Image and Vision Computing
Single view vision

Instructor: Chen Yisong
HCI & Multimedia Lab, Peking University

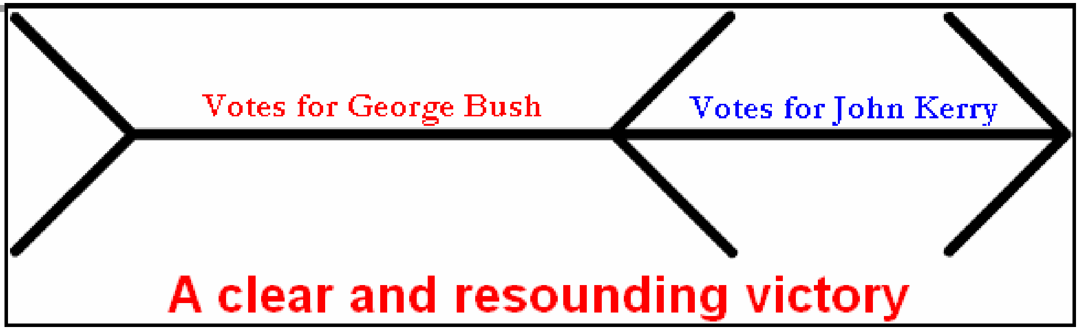
Projective geometry

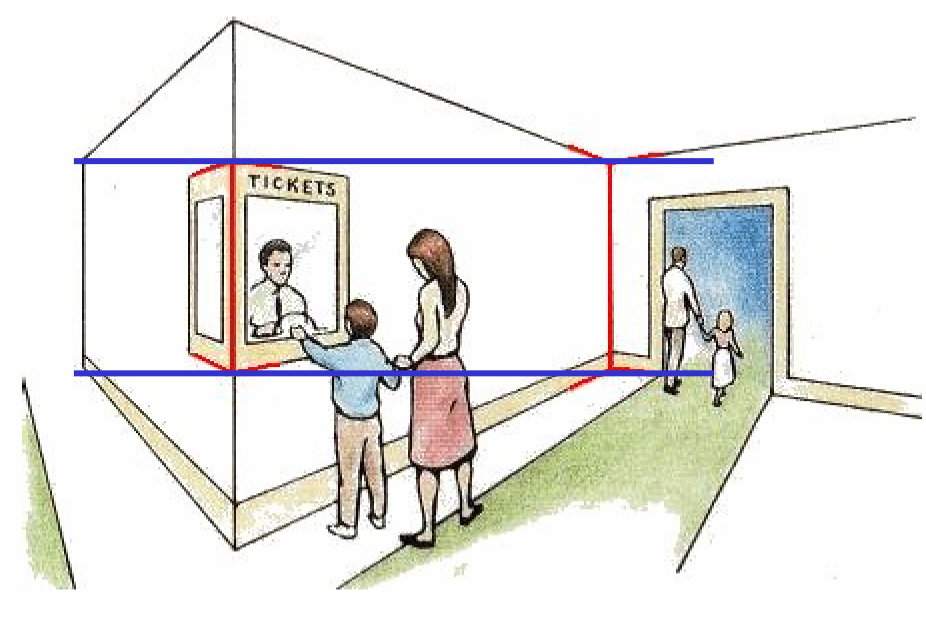
- How does a scene map to its image?
  - Projective Geometry
  - Homography

Ames Room

- Readings
  - Mundy, J.L. and Zisserman, A., Geometric Invariance in Computer Vision, Appendix: Projective Geometry for Machine Vision, MIT Press, Cambridge, MA, 1992
  - available online: http://www.cs.cmu.edu/~ph/869/papers/zisser-mundy.pdf

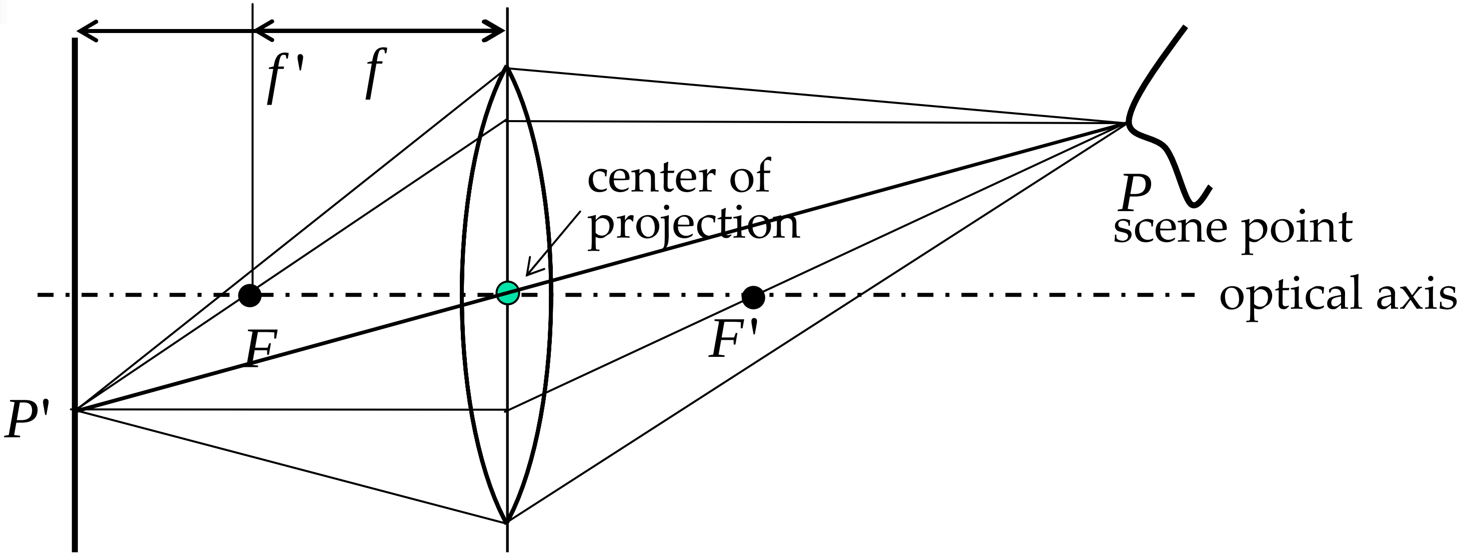
Müller-Lyer Illusion





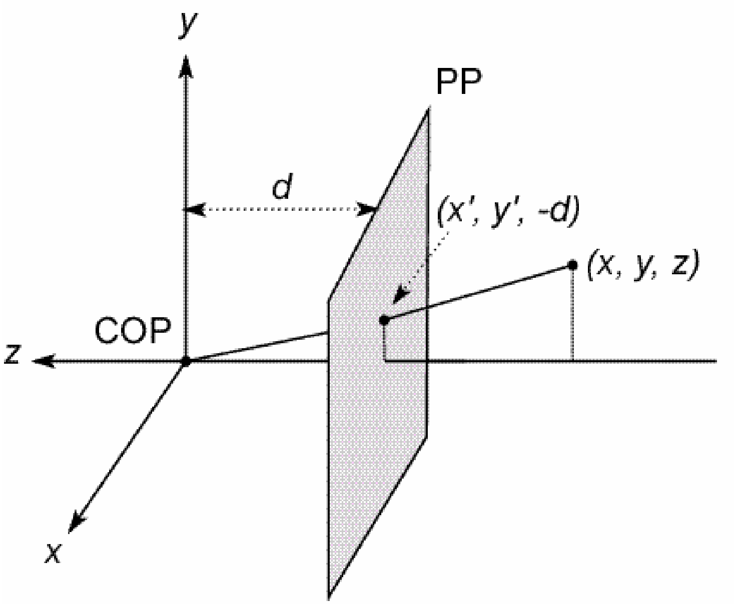
http://www.michaelbach.de/ot/sze_muelue/index.html

Modeling Projection



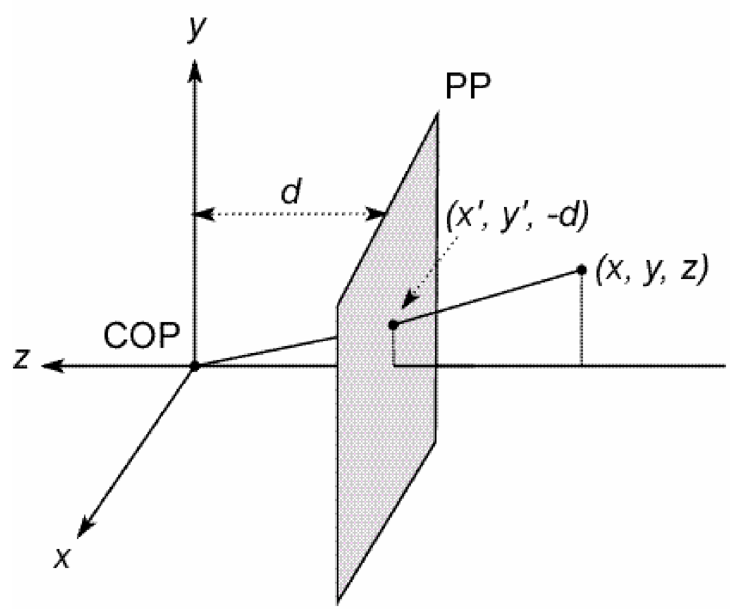
f': *effective* focal length (will be d from next slide)

Modeling Projection



- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP
  - The camera looks down the *negative* z axis
    - we need this if we want right-handed-coordinates

Modeling Projection



- Perspective Projection
  - Compute intersection with PP of ray from *(x,y,z)* to COP
  - Derived using similar triangles $(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d)$
  - We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

Homogeneous Coordinates

- Is this a linear transformation?
  - no—division by *z* is nonlinear $(x, y, z) \to (-d\frac{x}{z}, \ -d\frac{y}{z})$
- Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad\qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous image coordinates homogeneous scene coordinates

- Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad\qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Perspective Projection

- Projection is a matrix multiplication using **homogeneous coordinates**:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow \left(-d\frac{x}{z}, \quad -d\frac{y}{z}\right)$$

divide by third coordinate

- This is known as **perspective projection**
  - The matrix is the **projection matrix**

Perspective Projection

- How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

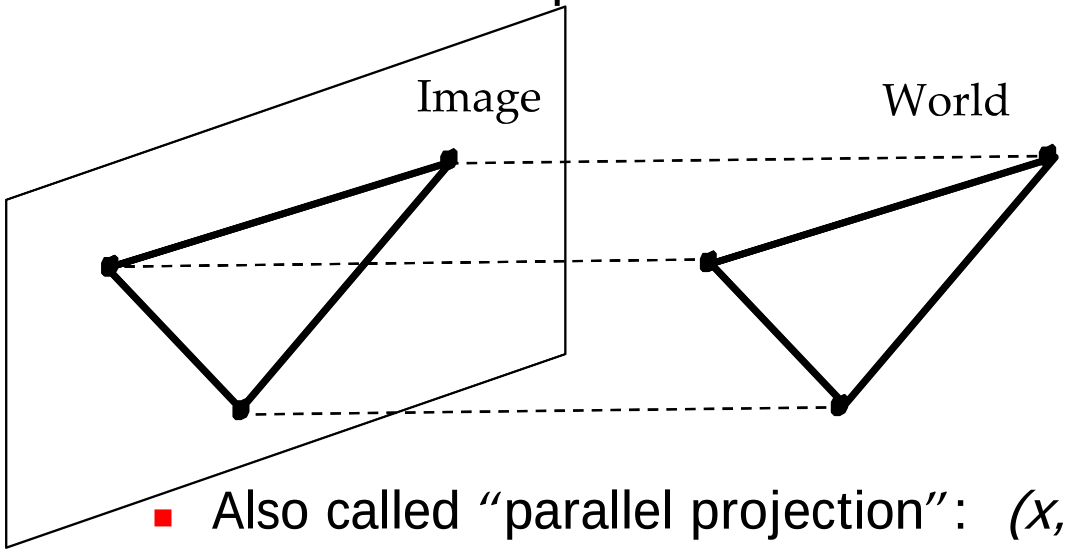
$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

- Conclusion: the Projection matrix is scale independent

Orthographic Projection

Special case of perspective projection

- Put effective optical center to infinite:



- Also called "parallel projection": *(x, y, z) → (x, y)*
- What's the projection matrix?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Other types of projection

- Scaled orthographic
  - Also called “weak perspective”

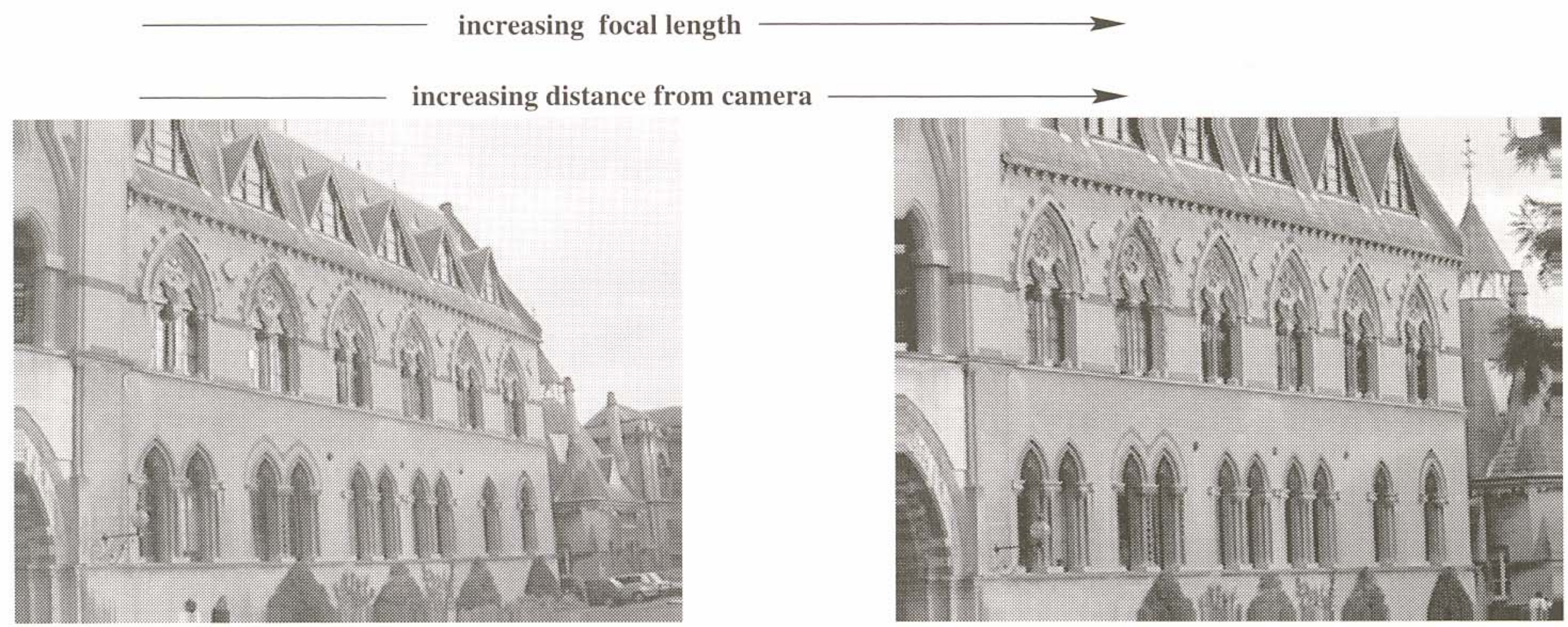
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

- Affine projection
  - Also called “paraperspective”

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Weak-Perspective Projeciton

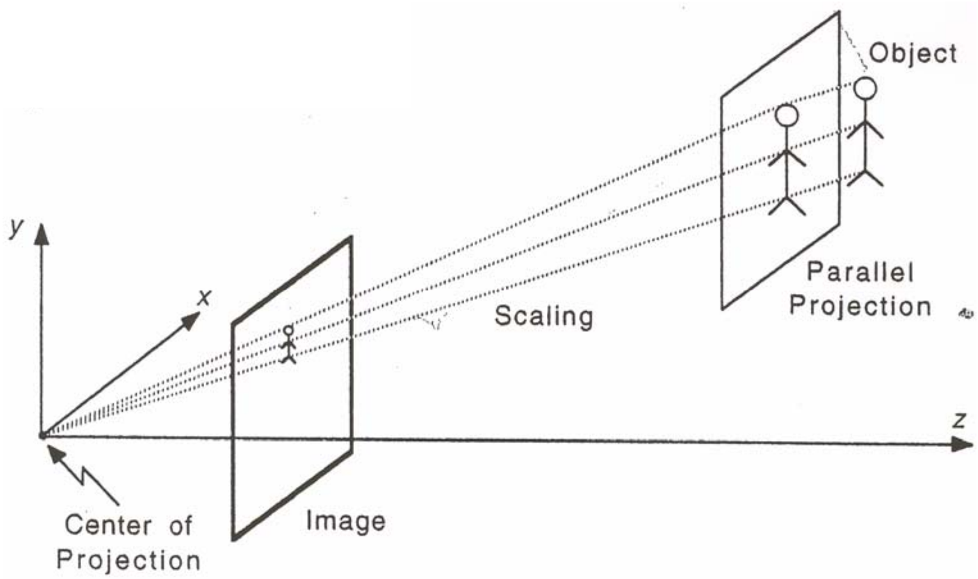
- Scaled orthographic



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

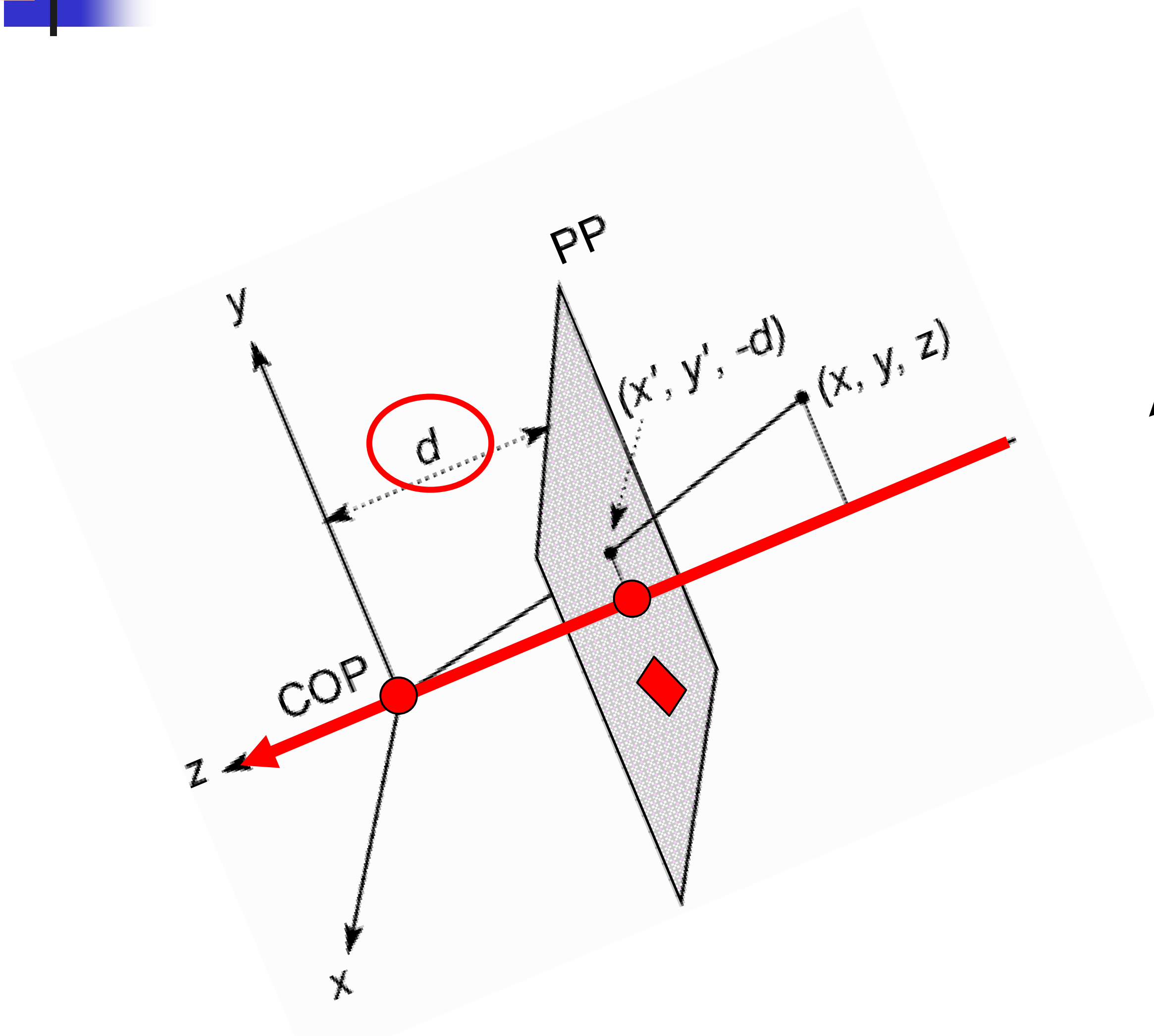
Affine Projection

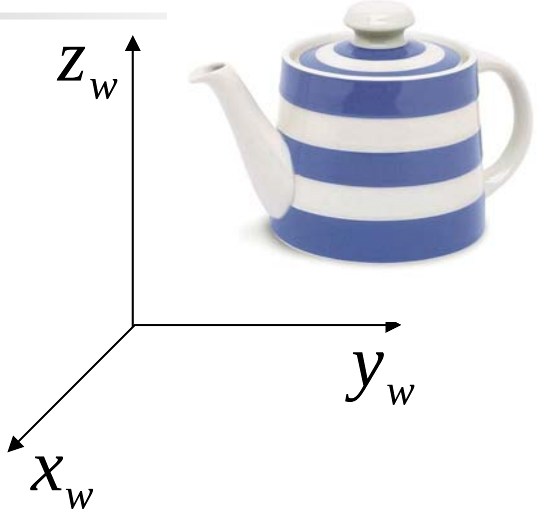
- Also called “paraperspective”



$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Camera Parameters





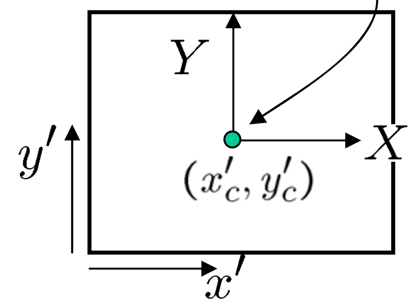
Camera parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"

Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

identity matrix

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{translation}}$$

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

Camera Calibration

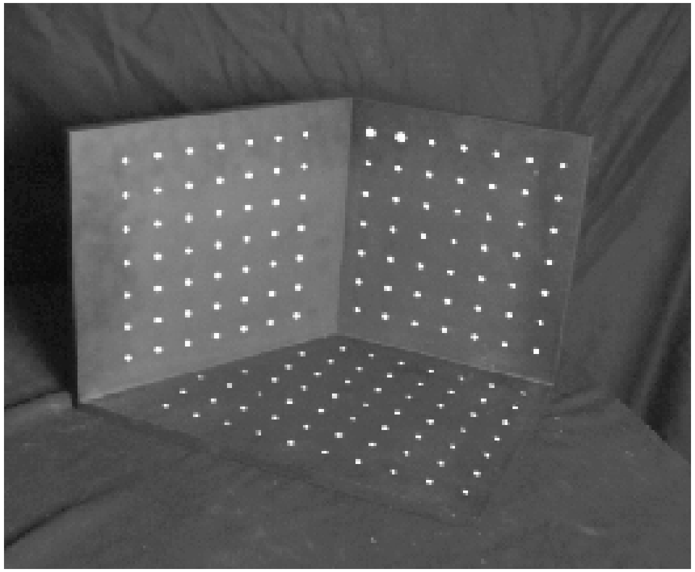
- Goal: estimate the camera parameters
  - Version 1: solve for projection matrix

$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$

  - Version 2: solve for camera parameters separately
    - intrinsics (focal length, principle point, pixel size)
    - extrinsics (rotation angles, translation)
    - radial distortion

Estimating the Projection Matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Issues

- must know geometry very accurately
- must know 3D->2D correspondence

Direct Linear Calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

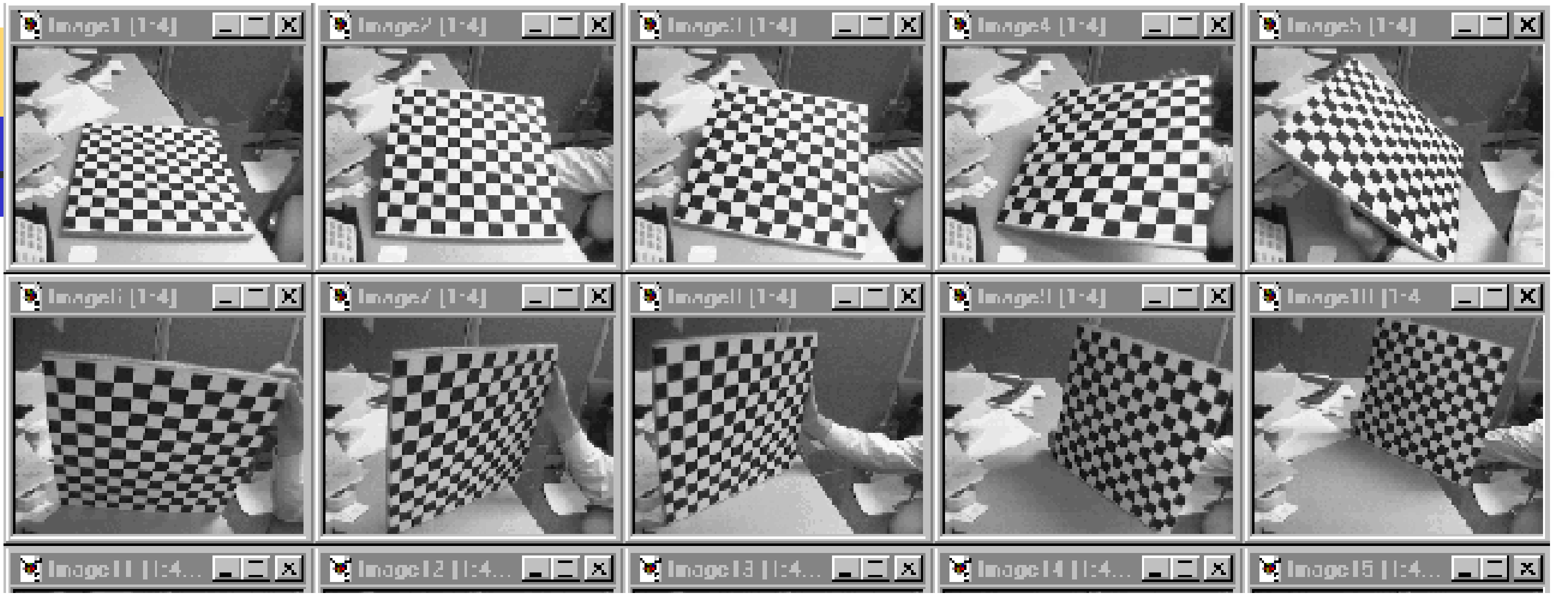
Direct linear calibration

- Advantage:
  - Very simple to formulate and solve
- Disadvantages:
  - Doesn’t tell you the camera parameters
  - Doesn’t model radial distortion
  - Hard to impose constraints (e.g., known focal length)
  - Doesn’t minimize the right error function

For these reasons, *nonlinear methods* are preferred

- Define error function E between projected 3D points and image positions
  - E is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize E using nonlinear optimization techniques
  - e.g., variants of Newton’s method (e.g., Levenberg Marquart)

Alternative: Multi-Plane Calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  - Intel's OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/

Chromaglyphs

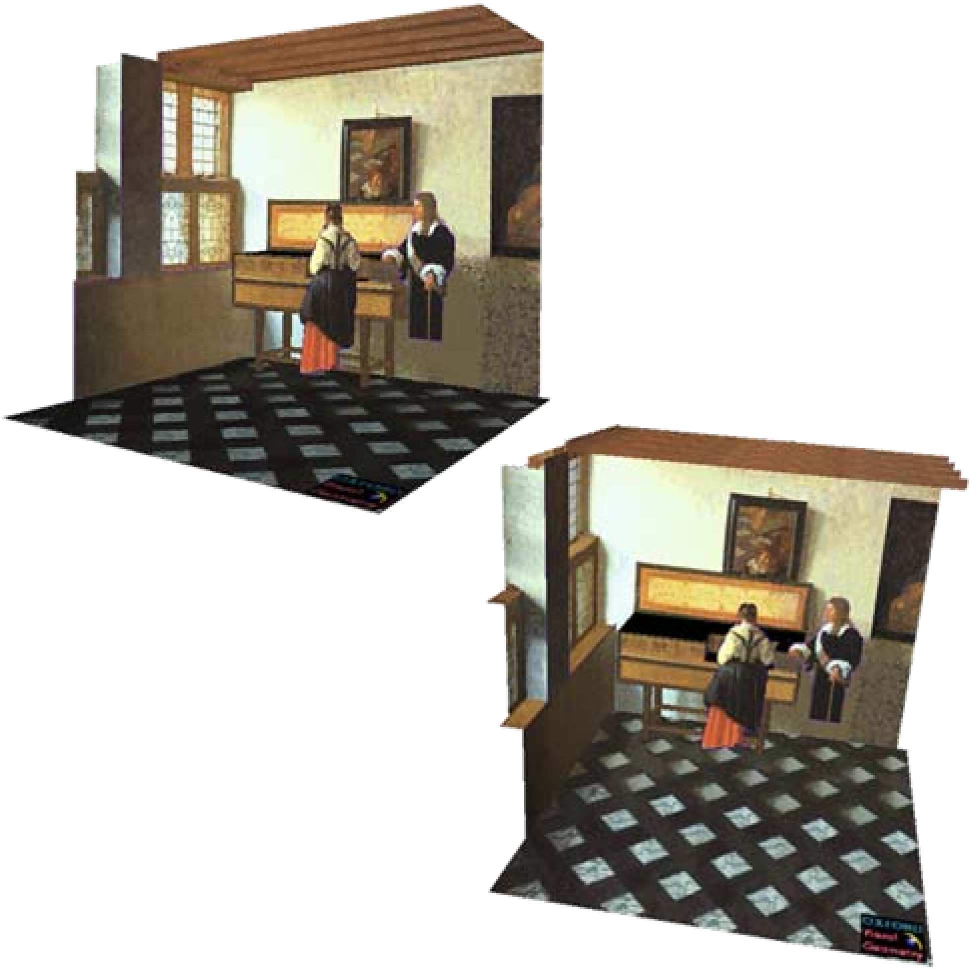
Courtesy of Bruce Culbertson, HP Labs
http://www.hpl.hp.com/personal/Bruce_Culbertson/ibr98/chromagl.htm

Projective geometry —what's it good for?

- Uses of projective geometry
  - Drawing
  - Measurements
  - Mathematics for projection
  - Undistorting images
  - Focus of expansion
  - Camera pose estimation, match move
  - Object recognition

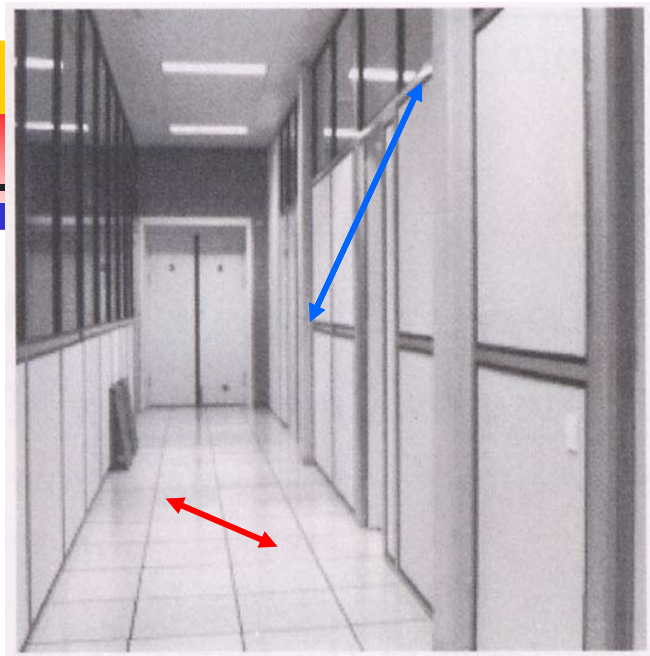
Applications of projective geometry

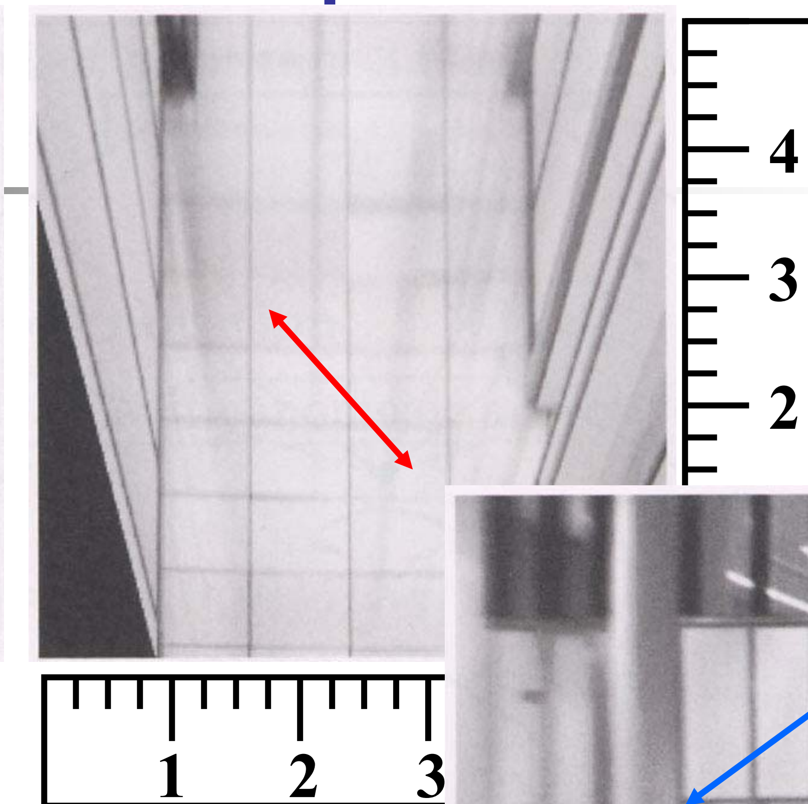
Vermeer's *Music Lesson*

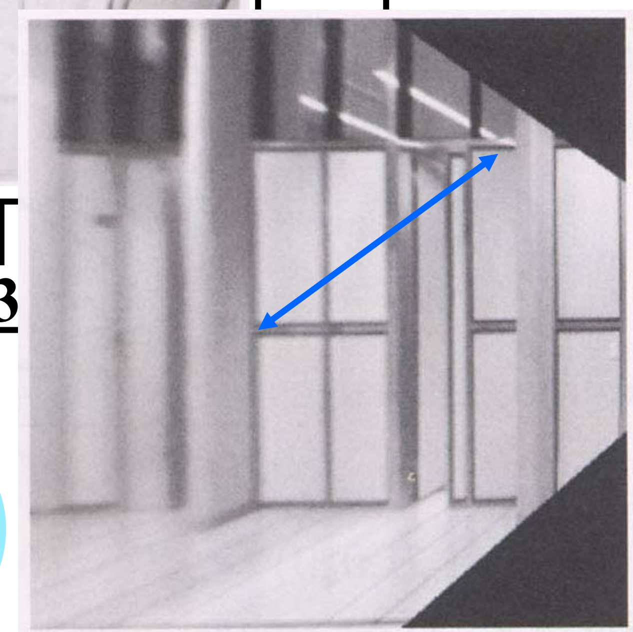


Reconstructions by Criminisi et al.

Measurements on planes



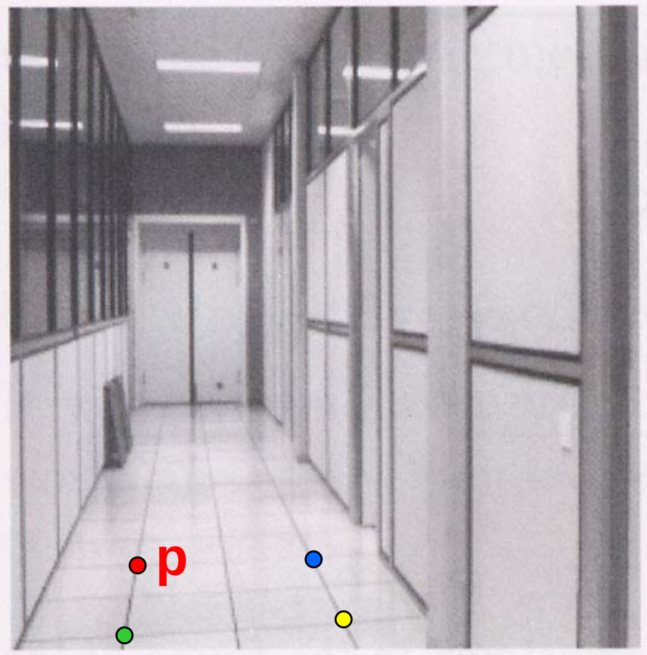


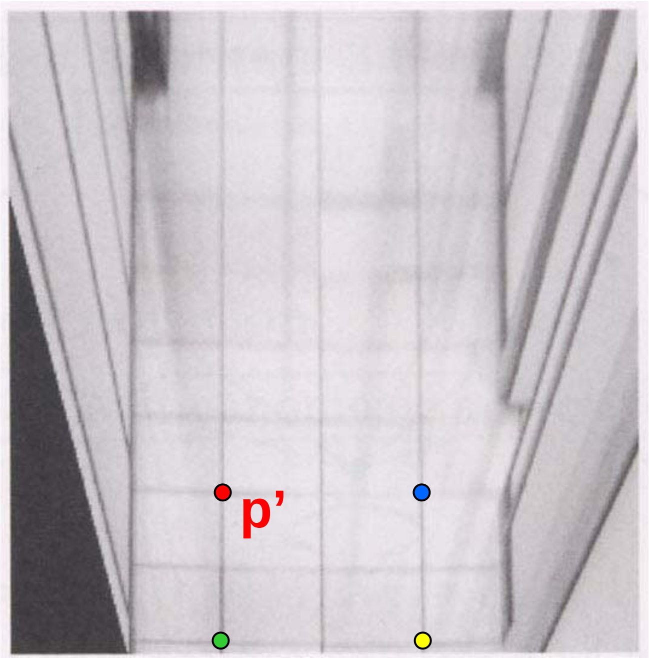


Approach: unwarp then measure

What kind of warp is this?

Image rectification





To unwarp (rectify) an image

- solve for homography **H** given **p** and **p'**
- solve equations of the form: w**p'** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

- Defines a least squares problem: $\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$
  - Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
  - Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
  - Works with 4 or more points

3D to 2D: "perspective" projection

- Matrix Projection:

$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

- What is *not* preserved under perspective projection?
- What IS preserved?

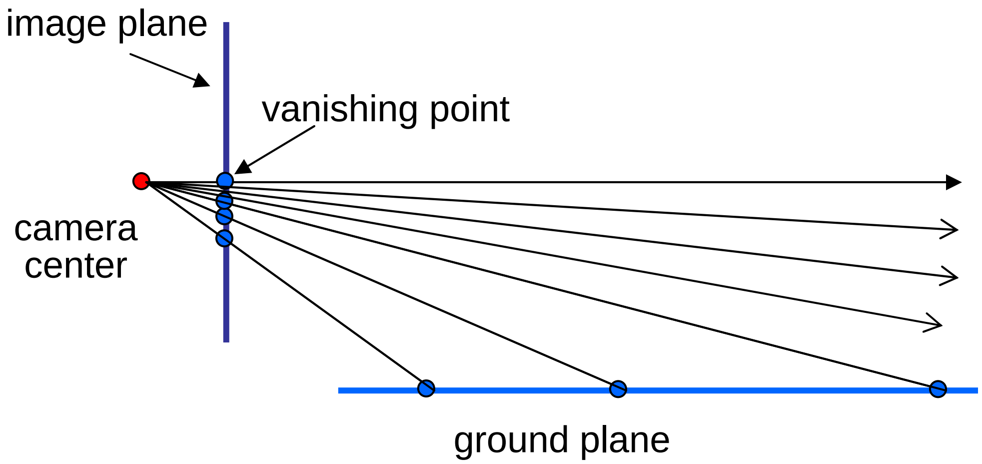
Homographies of points and lines

- Computed by 3x3 matrix multiplication
  - To transform a point: $\mathbf{p'} = \mathbf{Hp}$
  - To transform a line: $\mathbf{lp}=0 \rightarrow \mathbf{l'p'}=0$
    - $0 = \mathbf{lp} = \mathbf{lH^{-1}Hp} = \mathbf{lH^{-1}p'} \Rightarrow \mathbf{l' = lH^{-1}}$
    - lines are transformed by postmultiplication of $\mathbf{H^{-1}}$

3D projective geometry

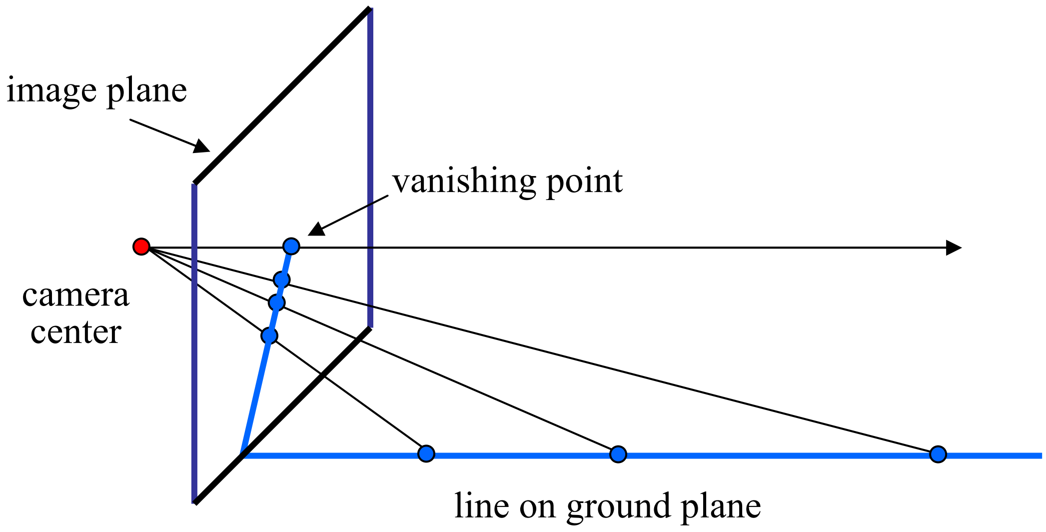
- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: $\mathbf{P} = (X,Y,Z,W)$
  - Duality
    - A plane $\mathbf{N}$ is also represented by a 4-vector
    - Points and planes are dual in 3D: $\mathbf{N}\,\mathbf{P}=0$
  - Projective transformations
    - Represented by 4x4 matrices $\mathbf{T}$: $\mathbf{P' = TP}$, $\mathbf{N' = N\,T^{-1}}$

Vanishing points

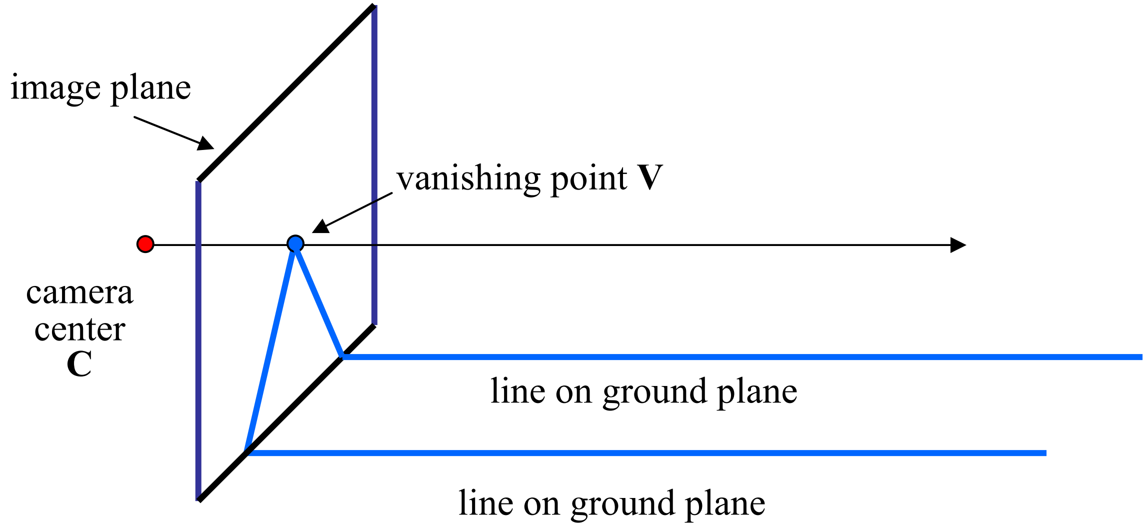


- Vanishing point
  - projection of a point at infinity

Vanishing points (2D)



Vanishing points

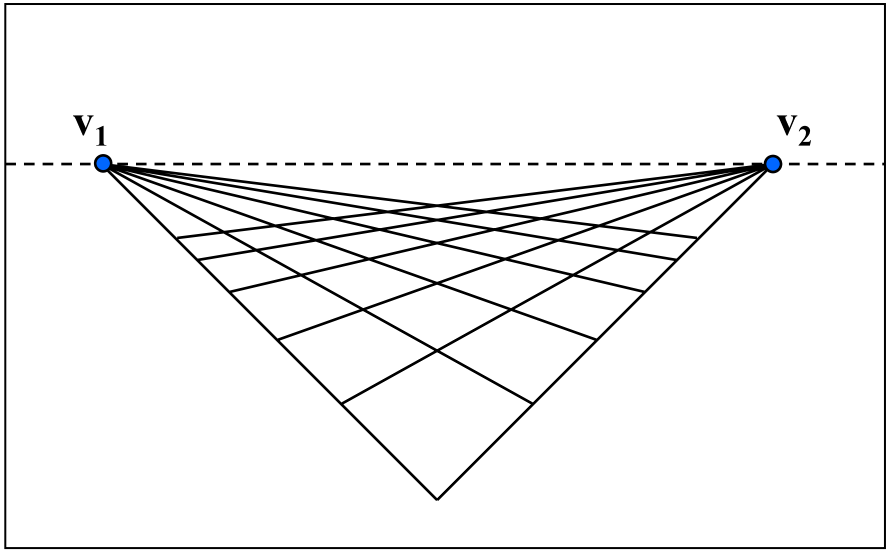


- Properties
  - Any two parallel lines have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact every pixel is a potential vanishing point

Vanishing points

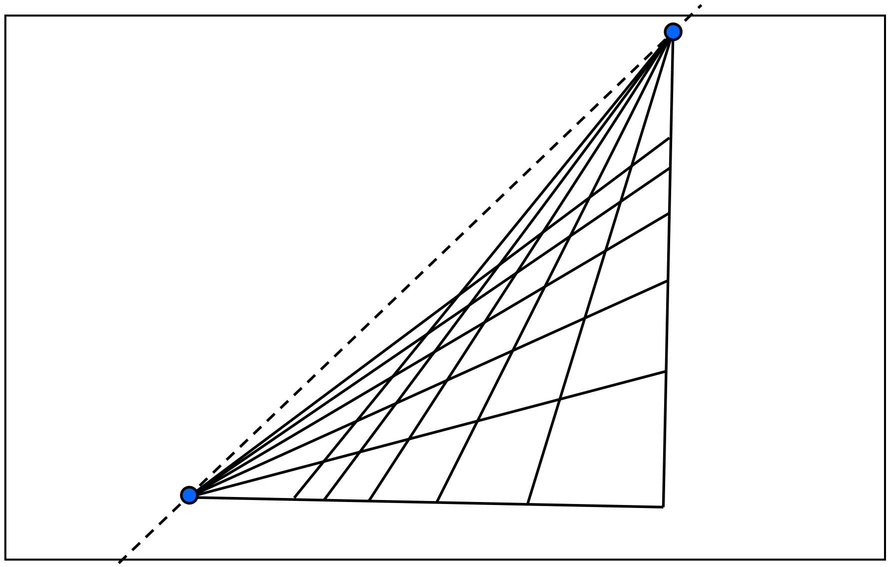
Image by Q-T. Luong (a vision researcher & photographer)

Vanishing lines



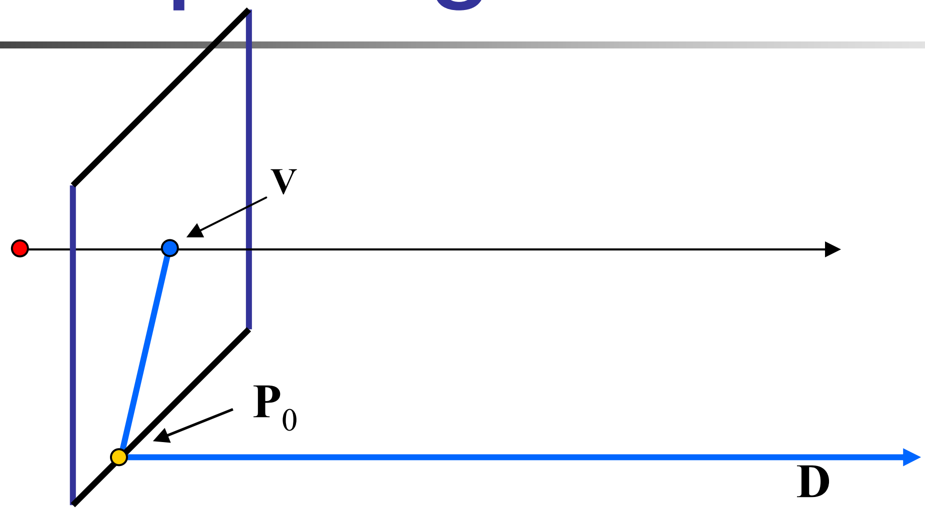
- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that *different* planes define *different* vanishing lines

Vanishing lines



- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes define different vanishing lines

Computing vanishing points

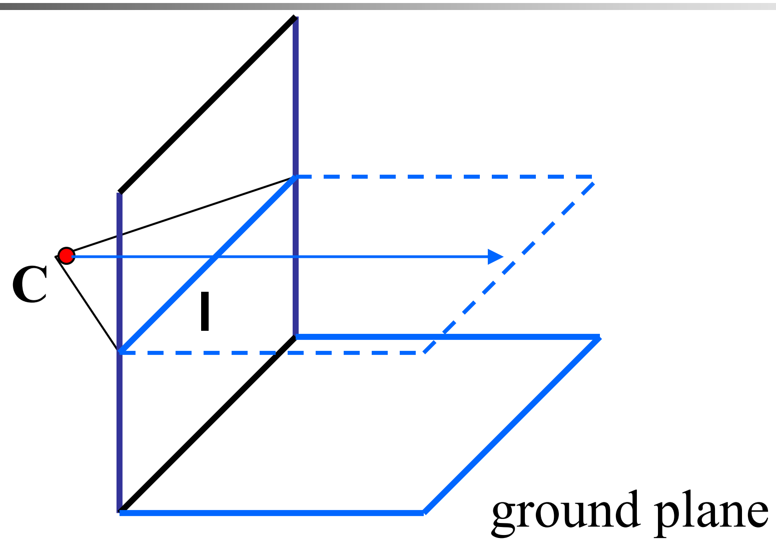


$$\mathbf{P} = \mathbf{P}_0 + t\mathbf{D}$$

$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X / t + D_X \\ P_Y / t + D_Y \\ P_Z / t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

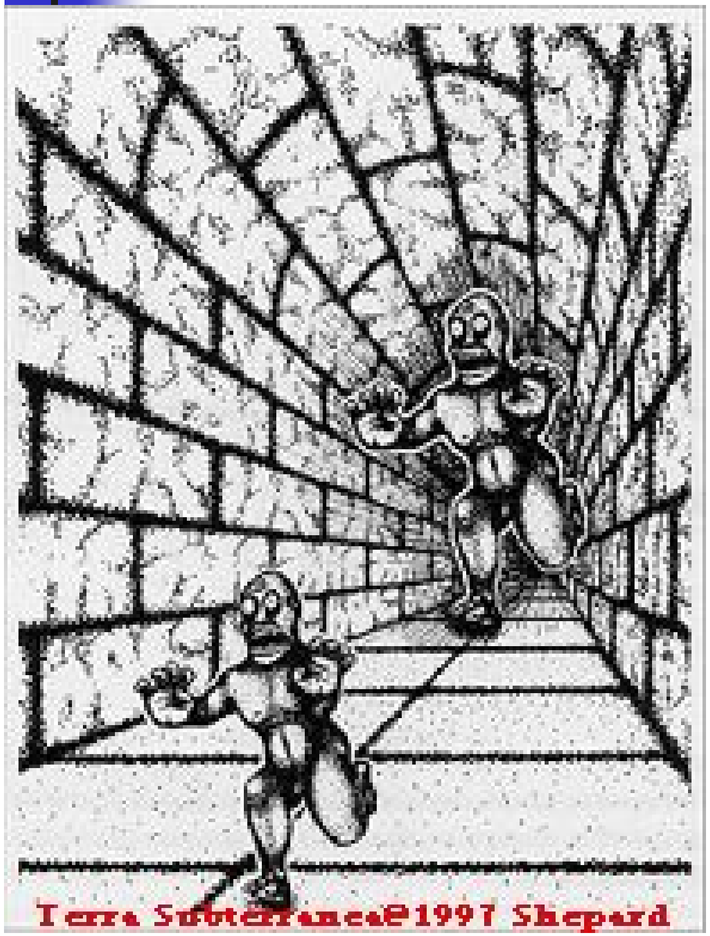
- Properties $\mathbf{v} = \mathbf{\Pi P}_\infty$
  - $\mathbf{P}_\infty$ is a point at *infinity*, **v** is its projection
  - They depend only on line *direction*
  - Parallel lines $\mathbf{P}_0$ + t**D**, $\mathbf{P}_1$ + t**D** intersect at $\mathbf{P}_\infty$

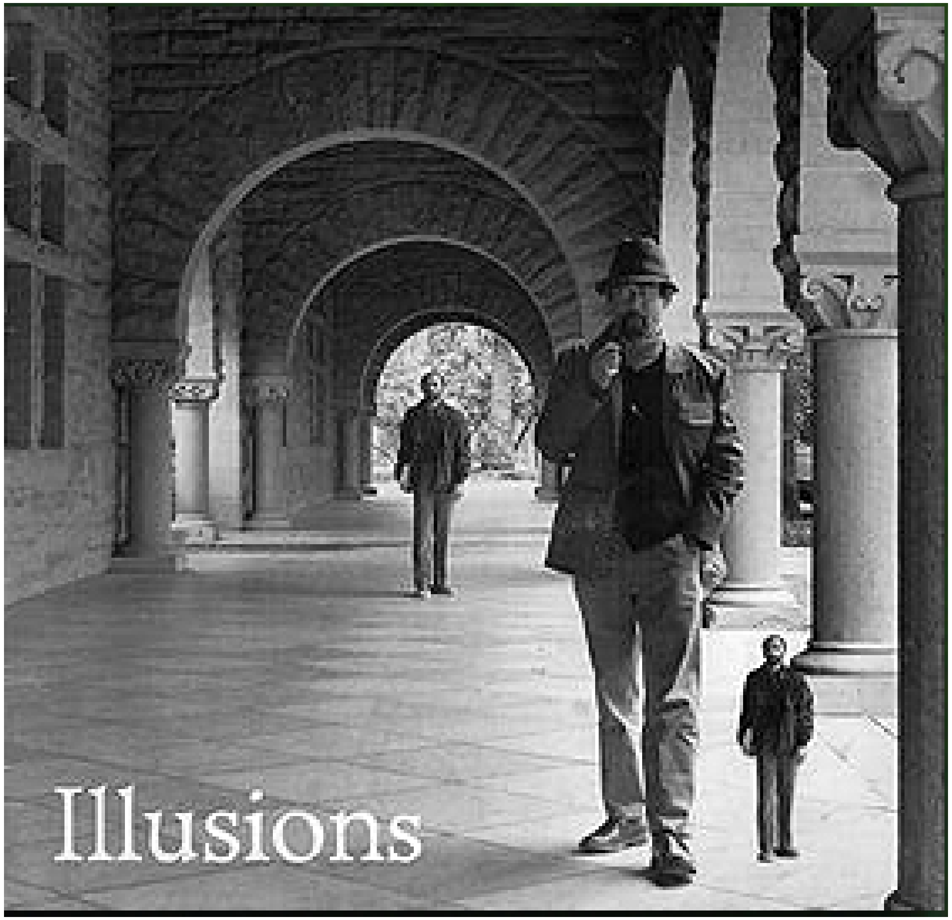
Computing vanishing lines



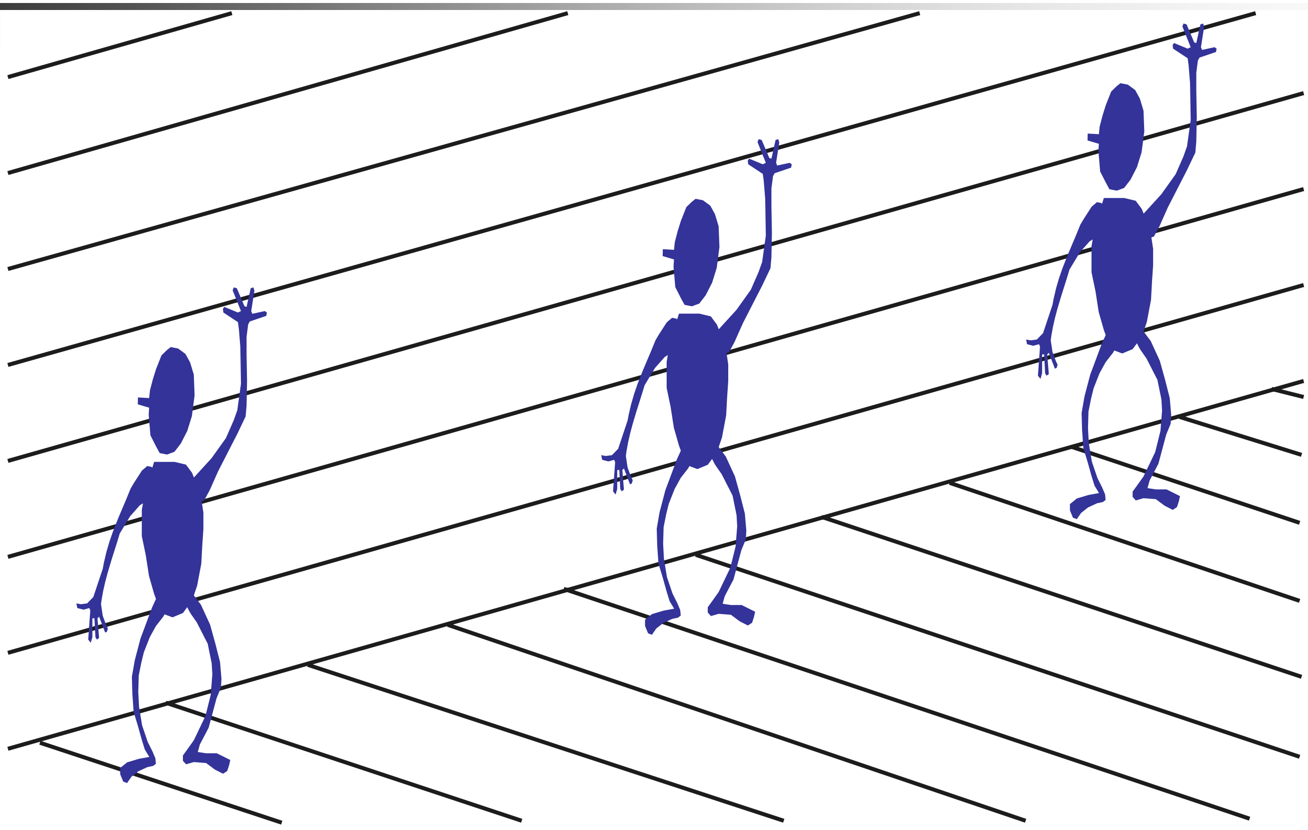
- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - Compute **l** from two sets of parallel lines on ground plane
  - All points at same height as **C** project to **l**
    - points higher than C project above l
  - Provides way of comparing height of objects in the scene

Fun with vanishing points

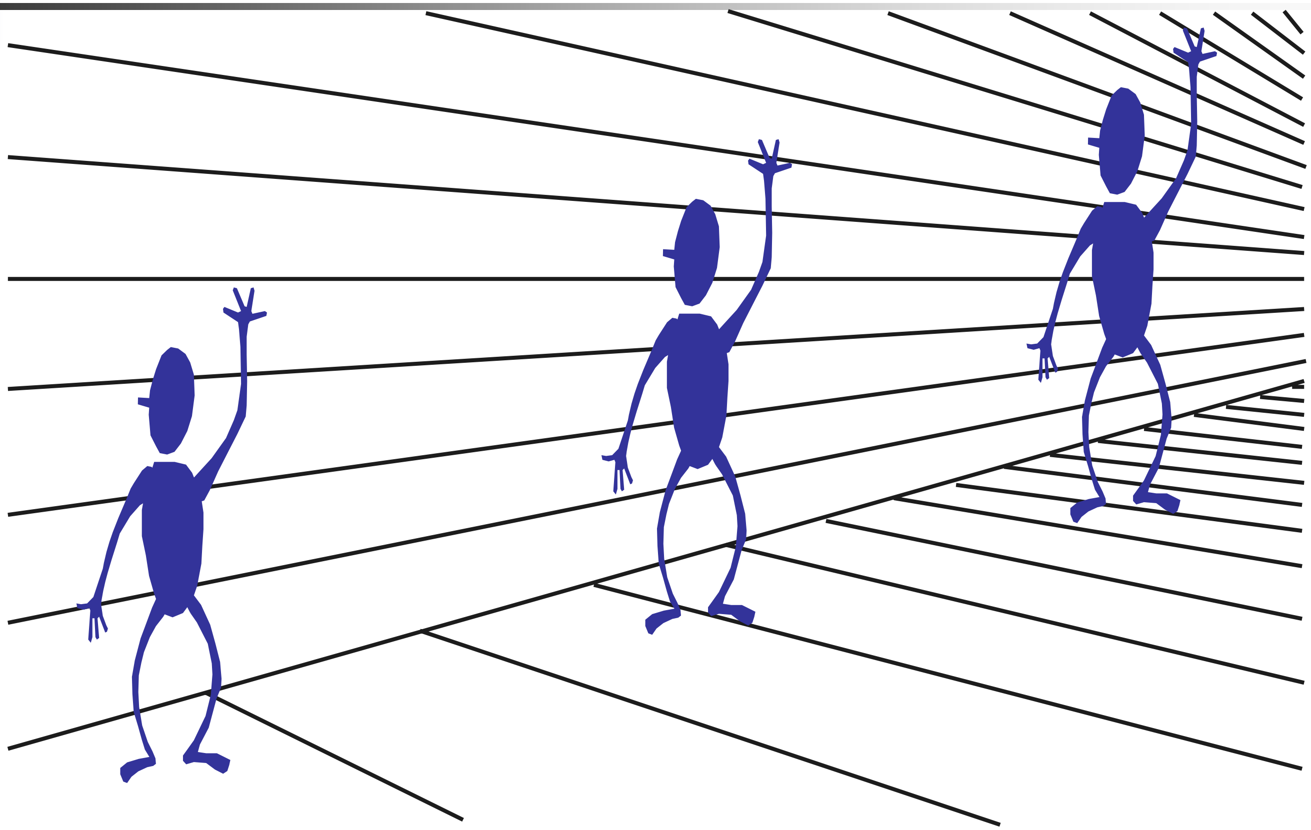




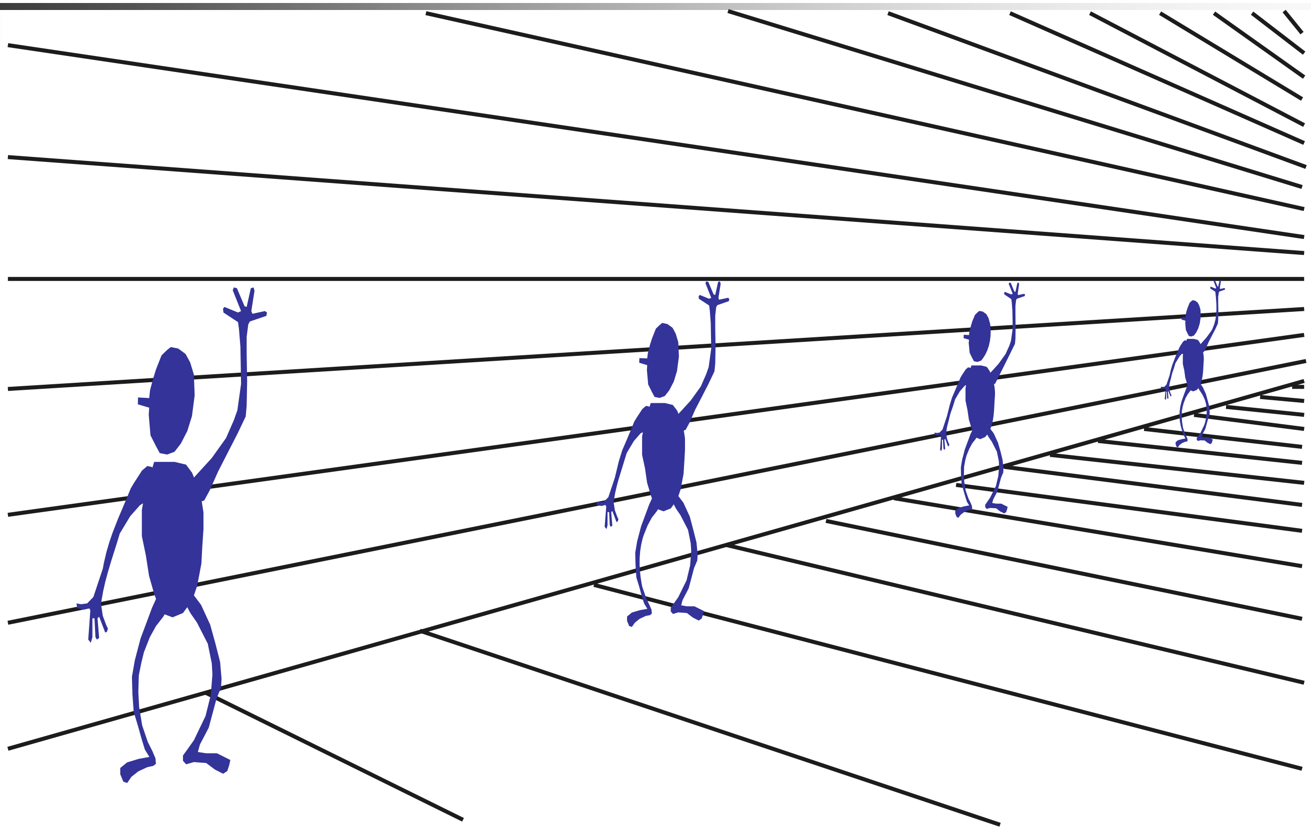
Perspective cues



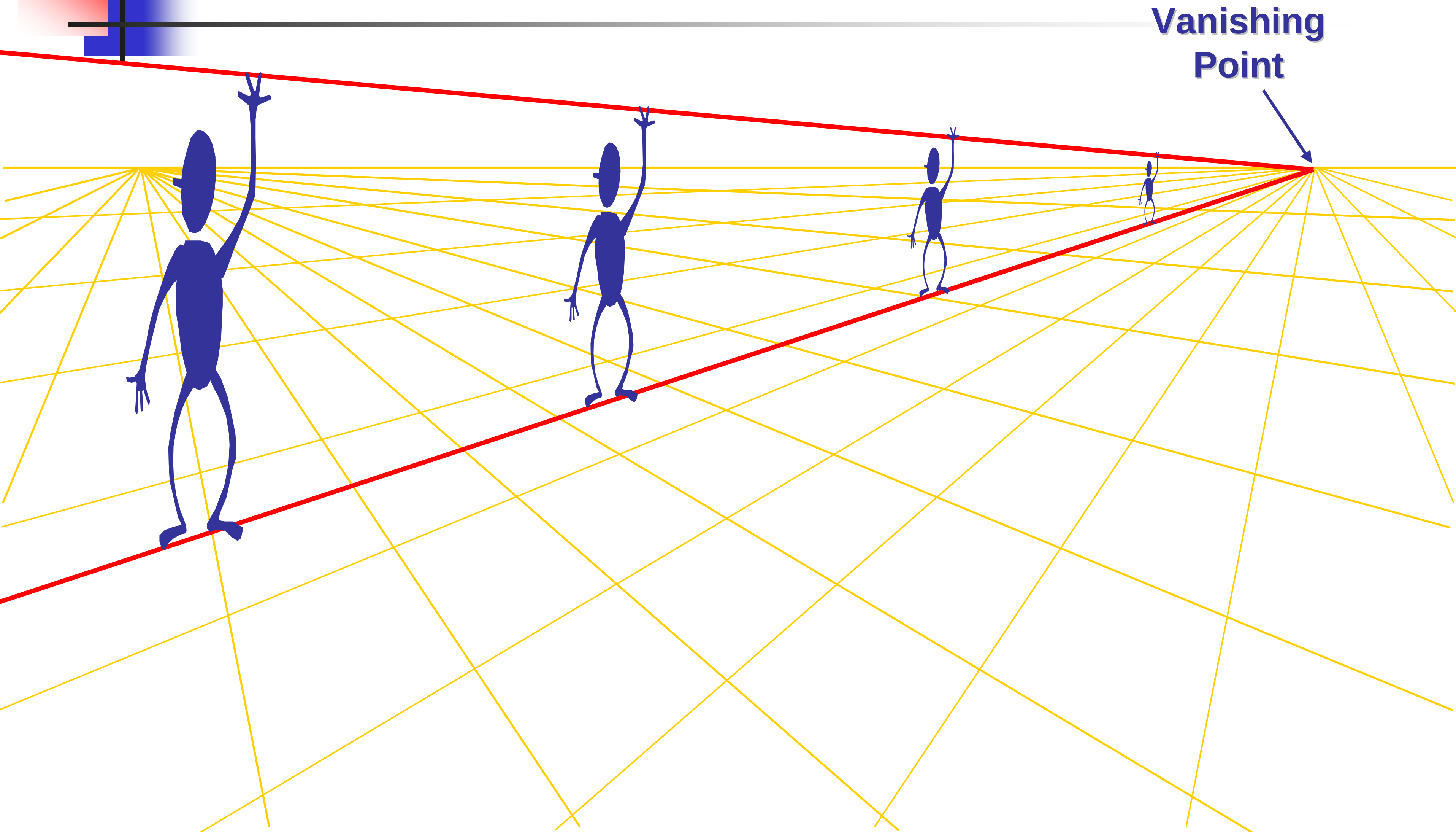
Perspective cues



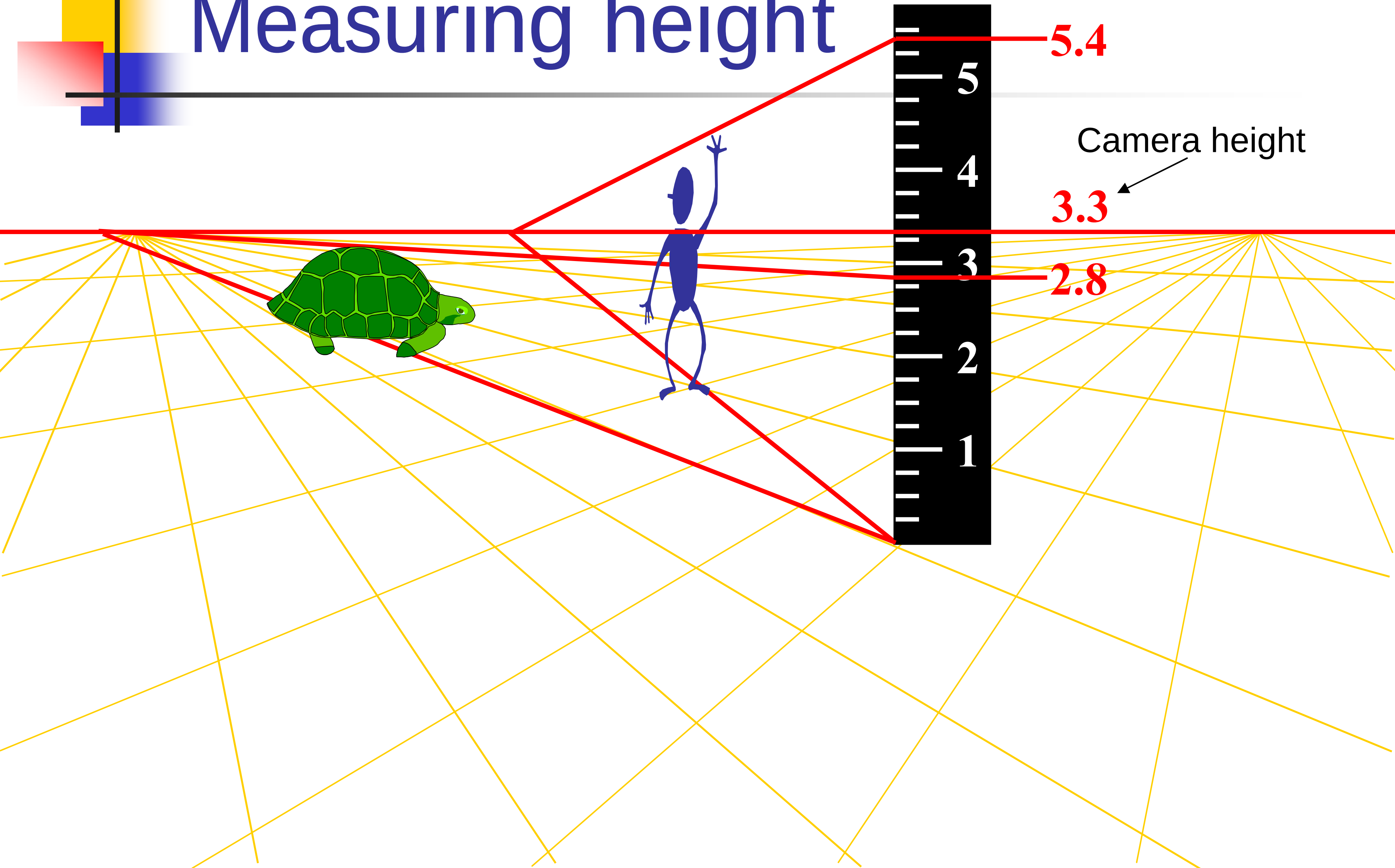
Perspective cues

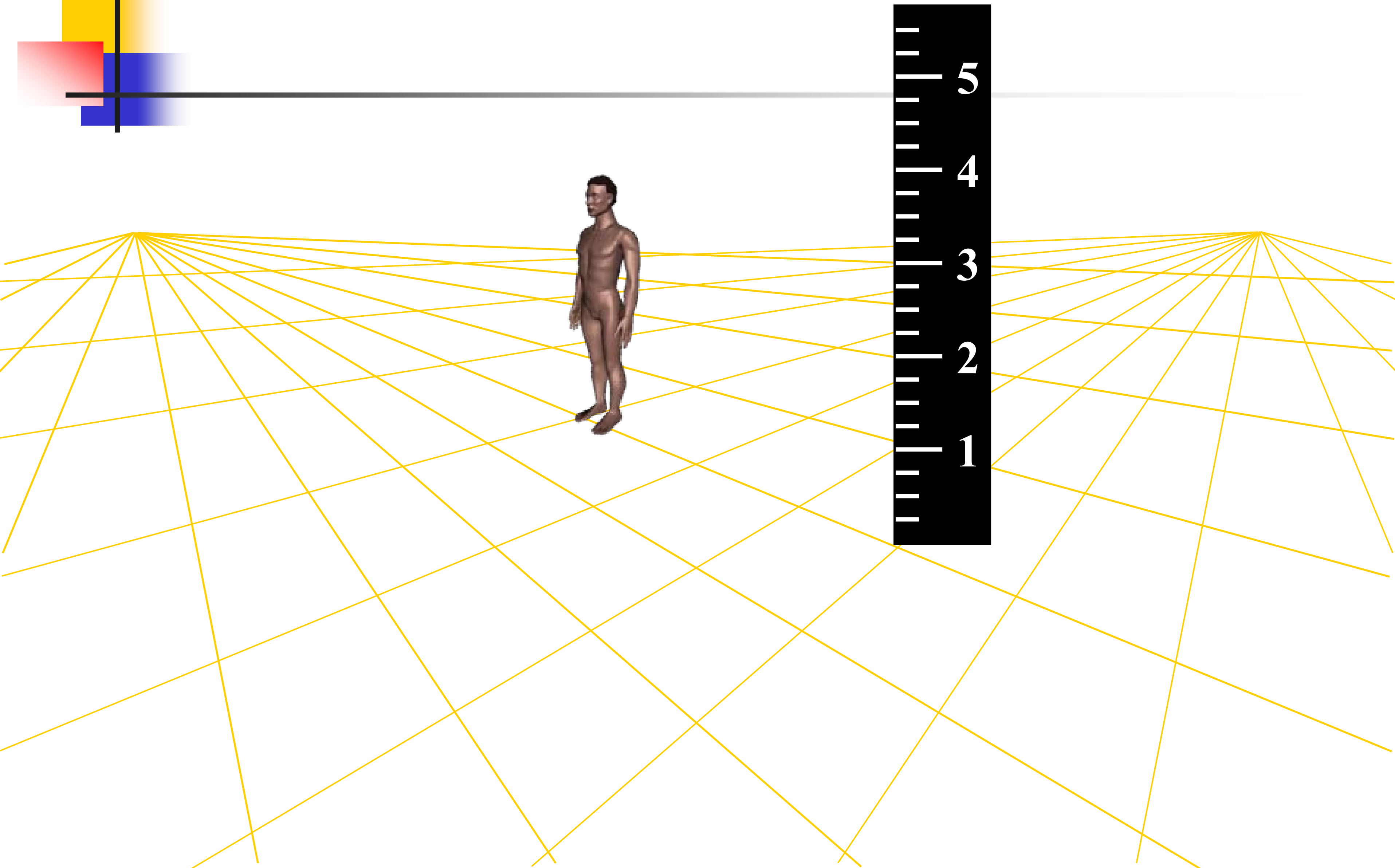


Comparing heights

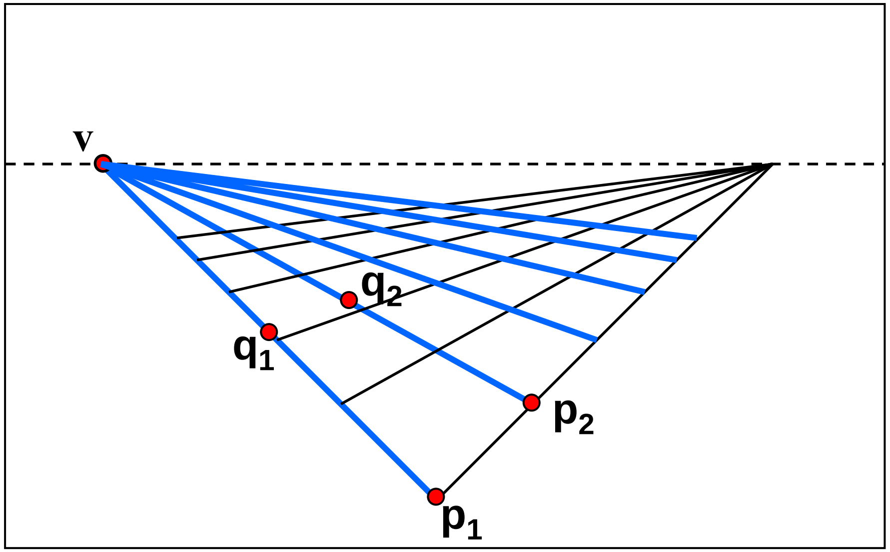


Measuring height




5
4
3
2
1

Computing vanishing points (from lines)

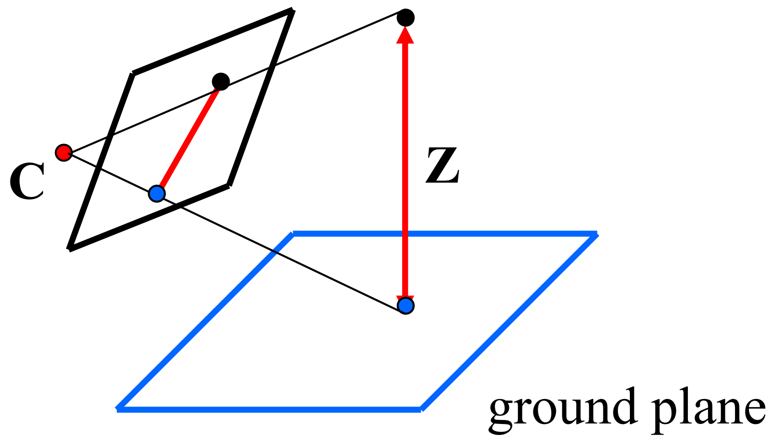


- Intersect p_1q_1 with p_2q_2

$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

- Least squares version
  - Better to use more than two lines and compute the "closest" point of intersection
  - See notes by [Bob Collins]for one good way of doing this:
    - http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt

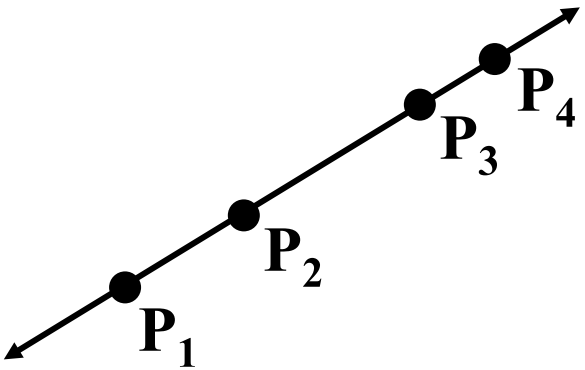
Measuring height without a ruler



- Compute Z from image measurements
  - Need more than vanishing points to do this

The cross ratio

- A Projective Invariant
  - Something that does not change under projective transformations (including perspective projection)



$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|}$$

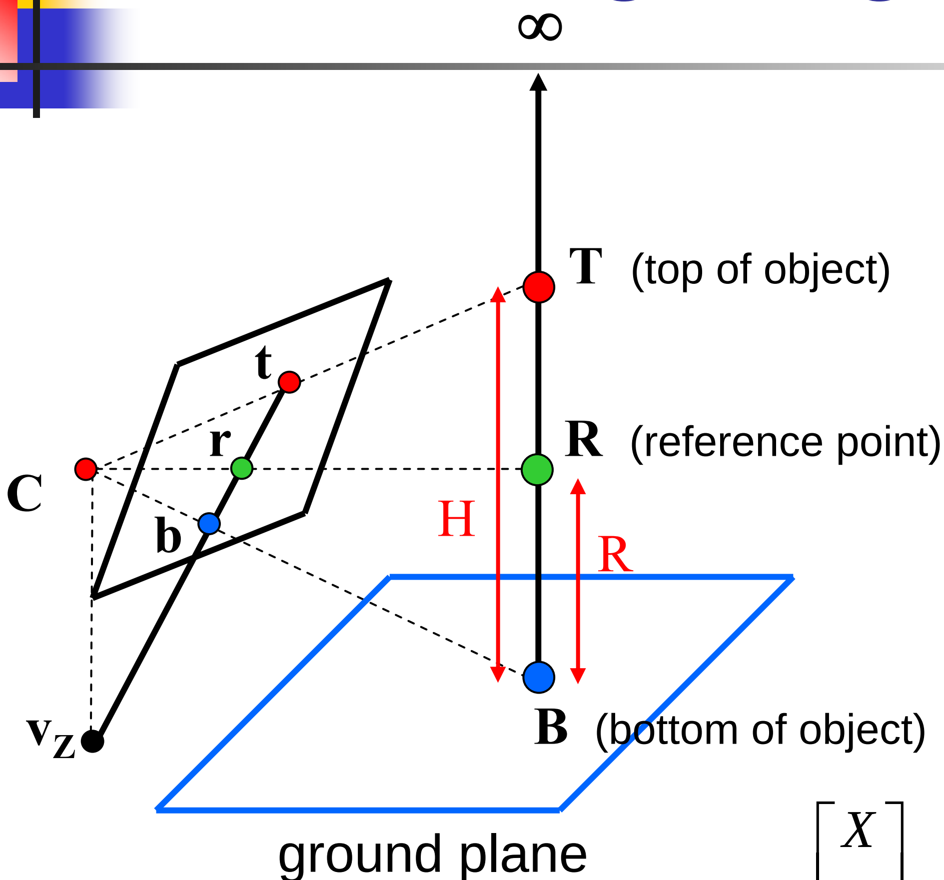
$$\mathbf{P}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

- Can permute the point ordering

$$\frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$$

  - 4! = 24 different orders (but only 6 distinct values)
- This is the fundamental invariant of projective geometry

Measuring height



$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|} = \frac{H}{R}$$

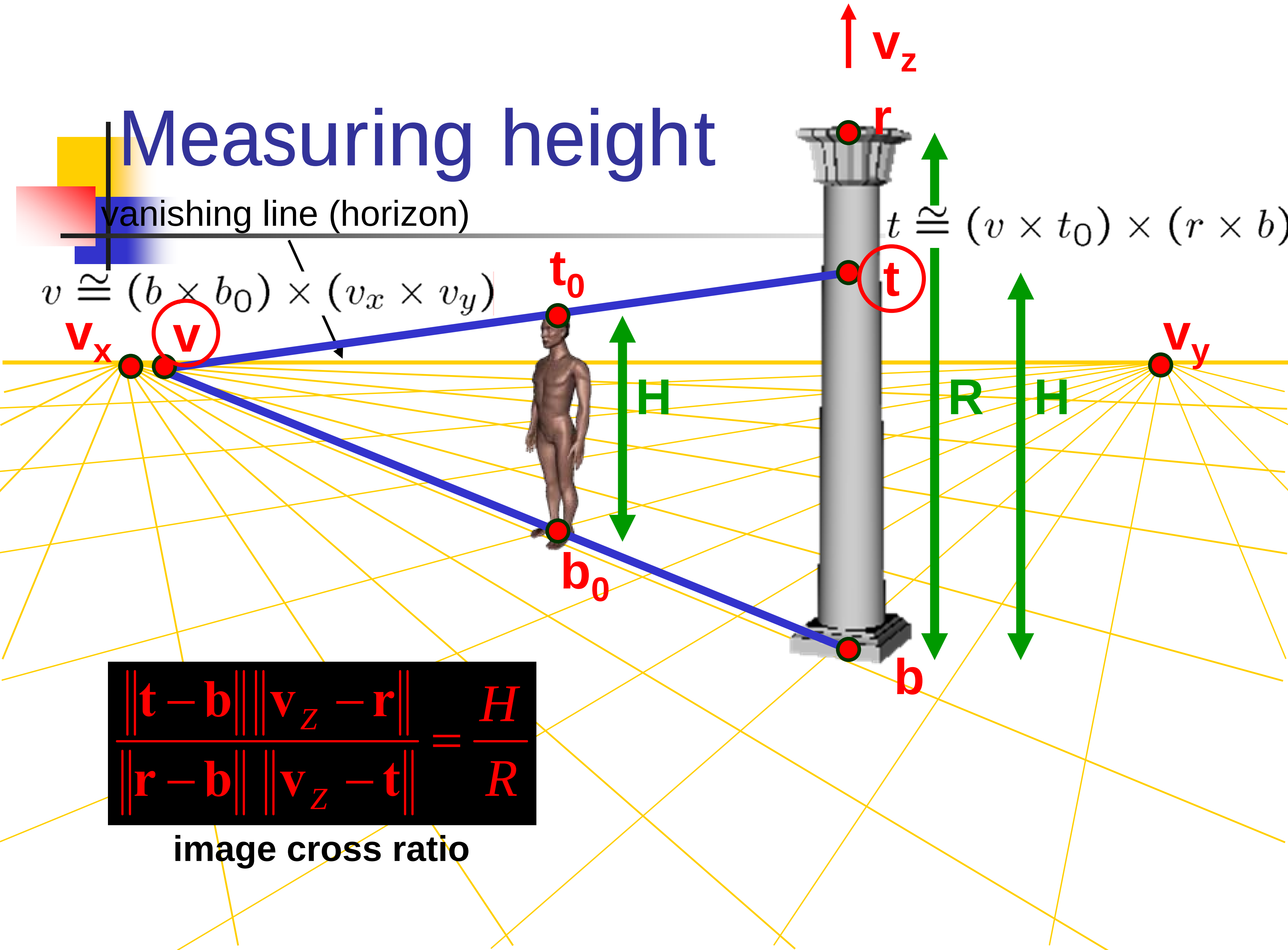
scene cross ratio

$$\frac{\|\mathbf{t}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|} = \frac{H}{R}$$

image cross ratio

scene points represented as $\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$ image points as $\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

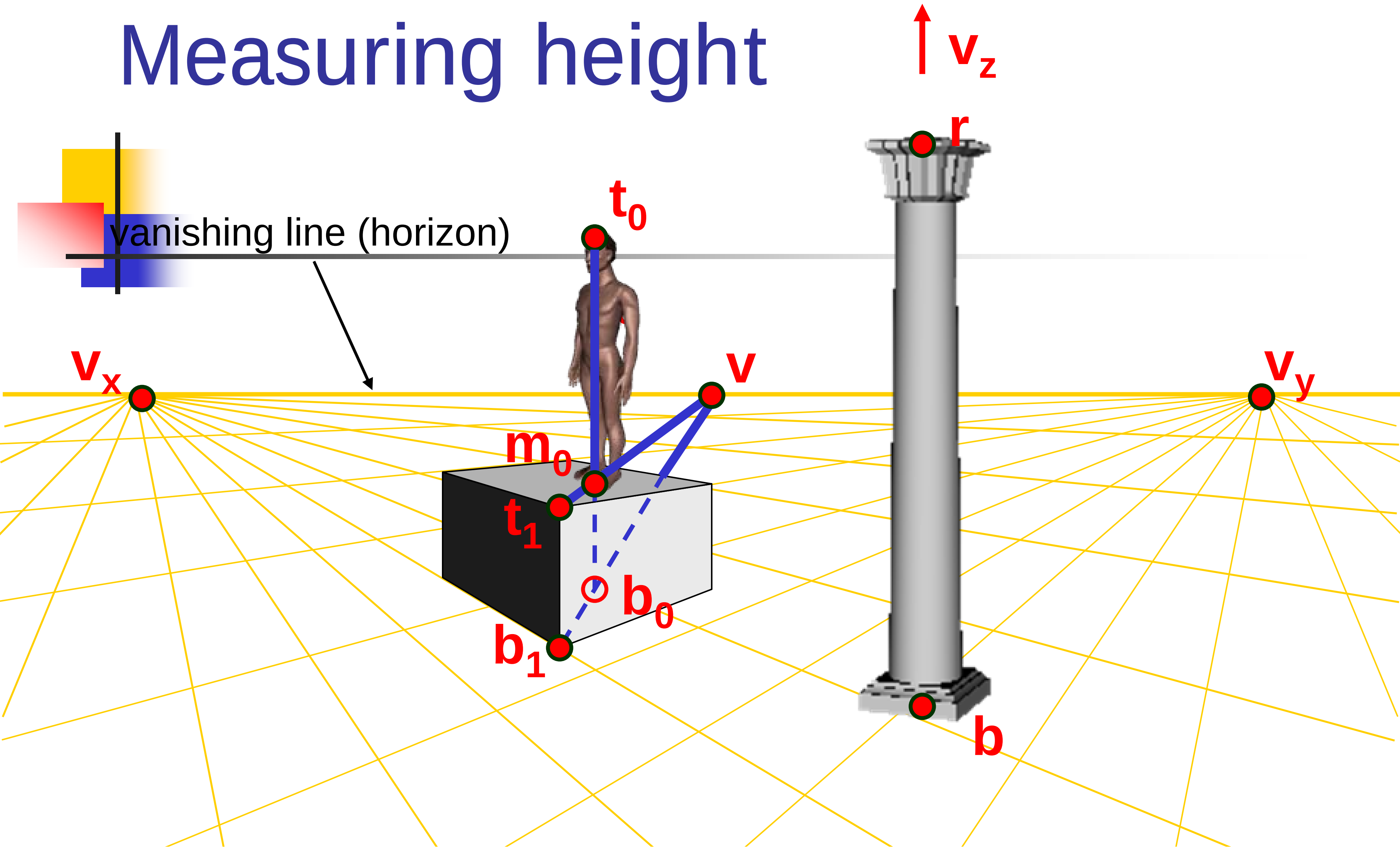
Measuring height



$$\frac{\|\mathbf{t}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|}=\frac{H}{R}$$

image cross ratio

Measuring height



- What if the point on the ground plane $\mathbf{b_0}$ is not known?
  - Here the guy is standing on the box, height of box is known
  - Use one side of the box to help find $\mathbf{b_0}$ as shown above

Computing (X,Y,Z) coordinates

- Okay, we know how to compute height (Z coords)
- how can we compute X, Y?

- Easy – Just map it to the reference plane

Vanishing point calibration

- Advantages:
  - only need to see vanishing points (e.g., architecture, table, …)
- Disadvantages:
  - not that accurate
  - need rectahedral object(s) in scene

Single View Metrology

- A. Criminisi, I. Reid and A. Zisserman (ICCV 99)
- Make scene measurements from a single image
  - Application: 3D from a single image
- Assumptions
  1. 3 orthogonal sets of parallel lines
  2. 4 known points on ground plane
  3. 1 height in the scene
- Can still get *affine reconstruction* without 2 and 3

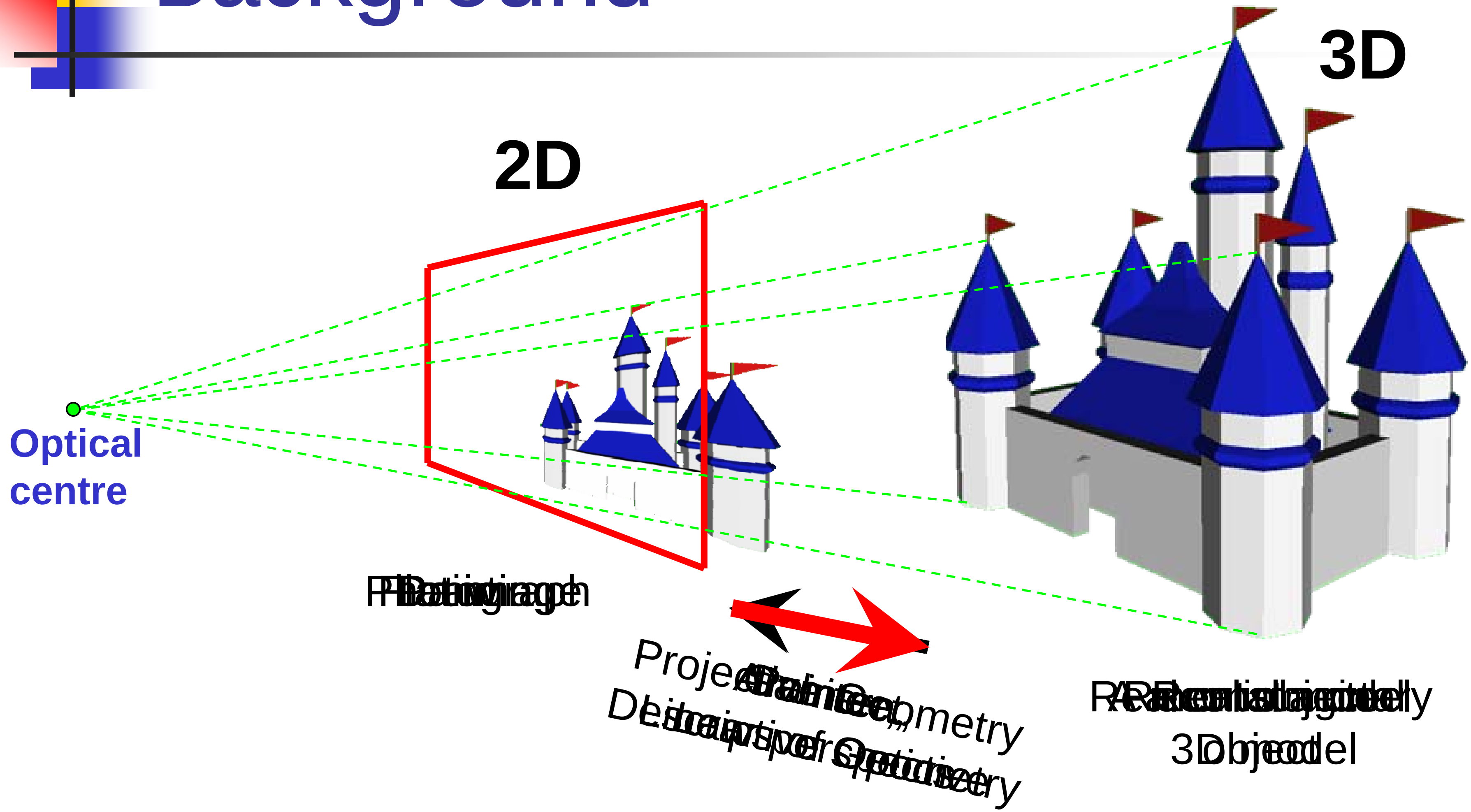
Problem

- Is it possible to extract 3D geometric information from single images?

YES

- How?
- Why?

Background

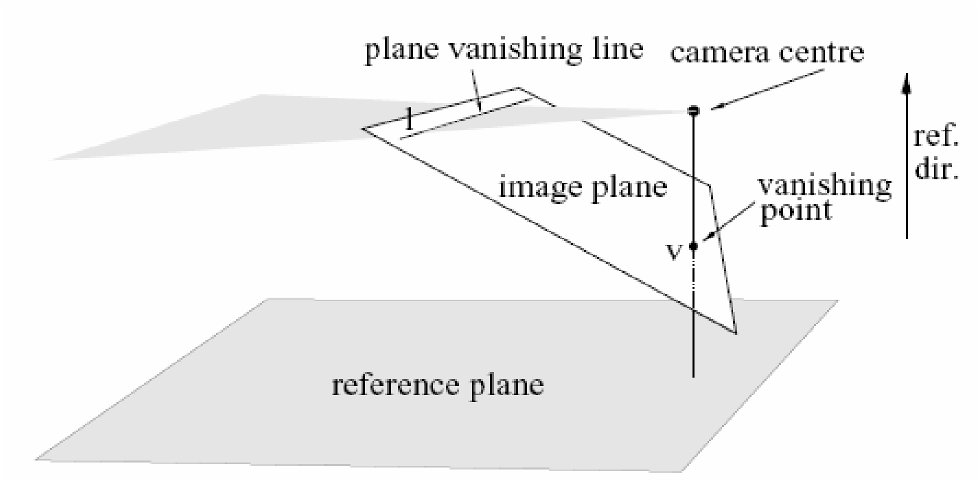


Introduction

- 3D affine measurements may be measured from a single perspective image

Geometry

- Overview
- Measurements between parallel lines
- Measurements on parallel planes
- Determining the camera position



Assumptions

- Assume that images are obtained by perspective projection
- Assume that, from the image, a:
  - <u>**vanishing line**</u> of a *reference plane*
  - <u>**vanishing point**</u> of another *reference direction*

may be determined from the image

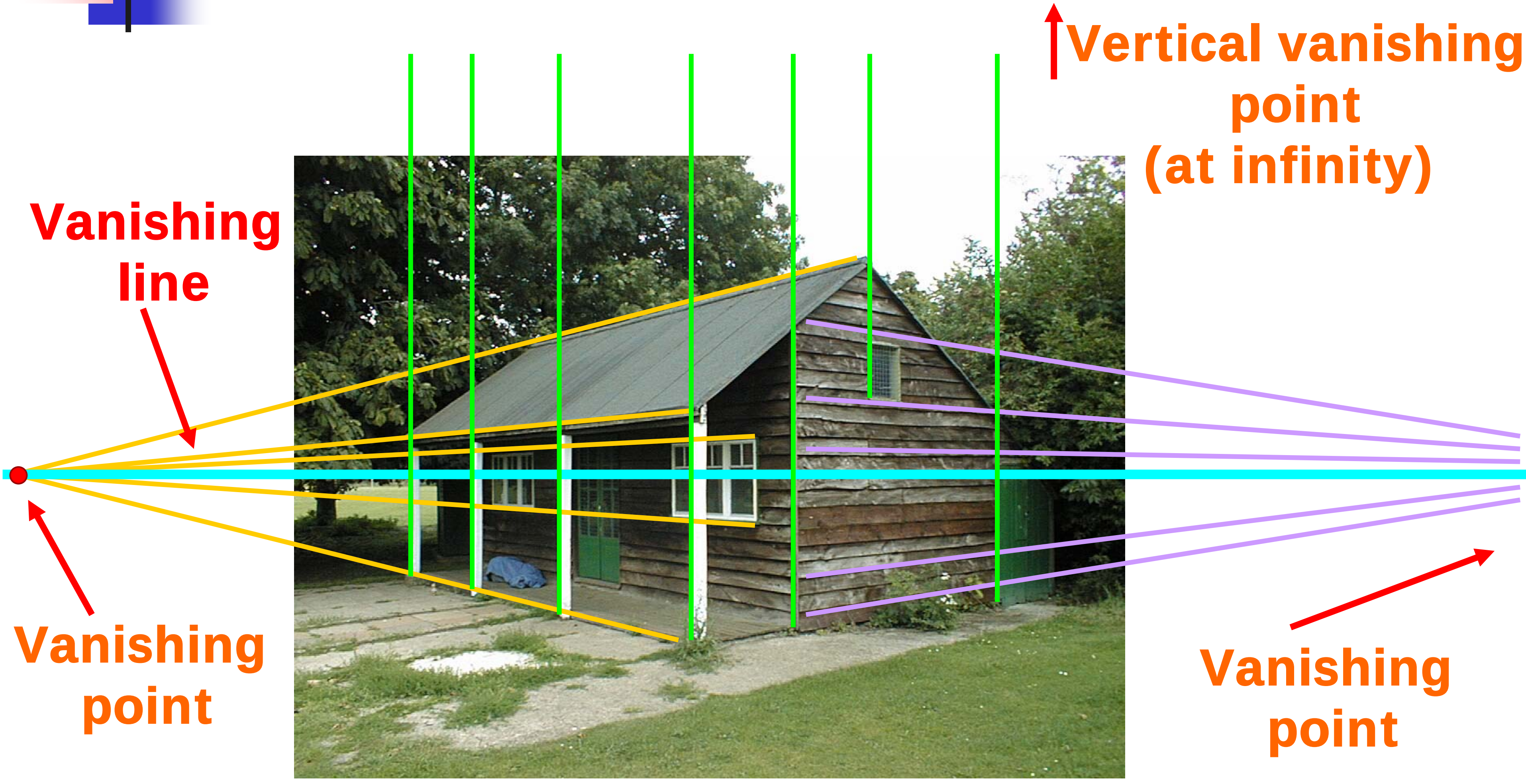
Geometric Cues

- Vanishing Line ℓ
  - Projection of the line at infinity of the reference plane into the image

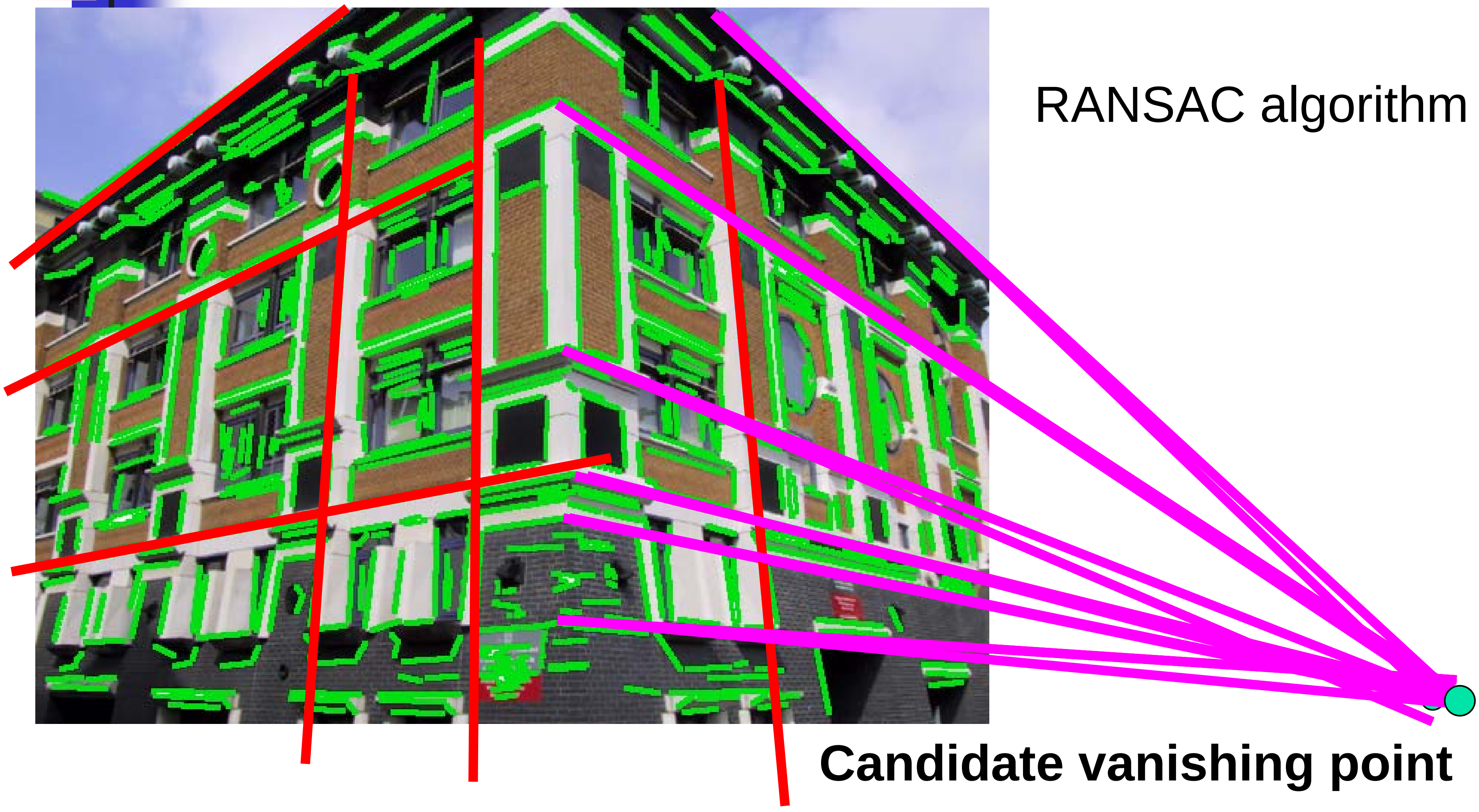
Geometric Cues

- Vanishing Point(s) v
  - A point at infinity in the reference direction
  - *Reference direction is NOT parallel to reference plane*
  - Also known as the vertical vanishing point

Geometric Cues



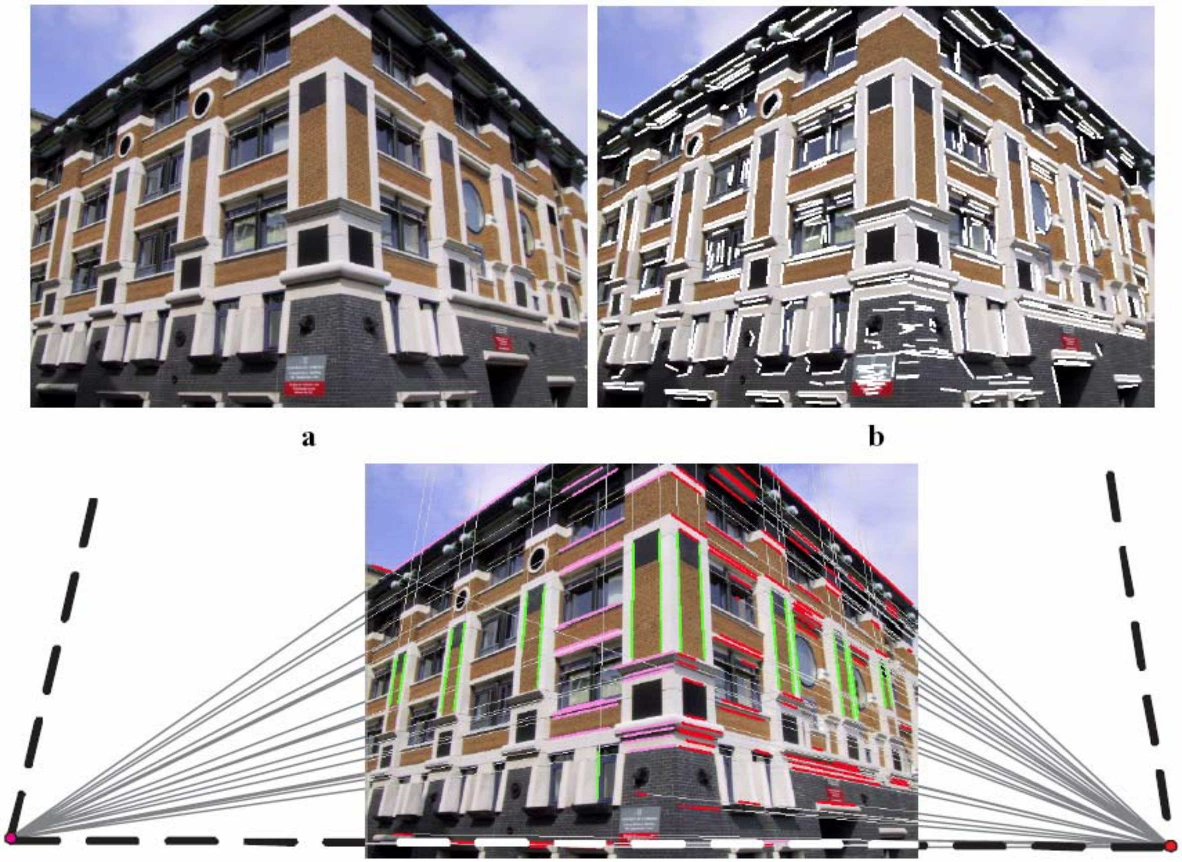
Automatic estimation of vanishing points and lines



RANSAC algorithm

Candidate vanishing point

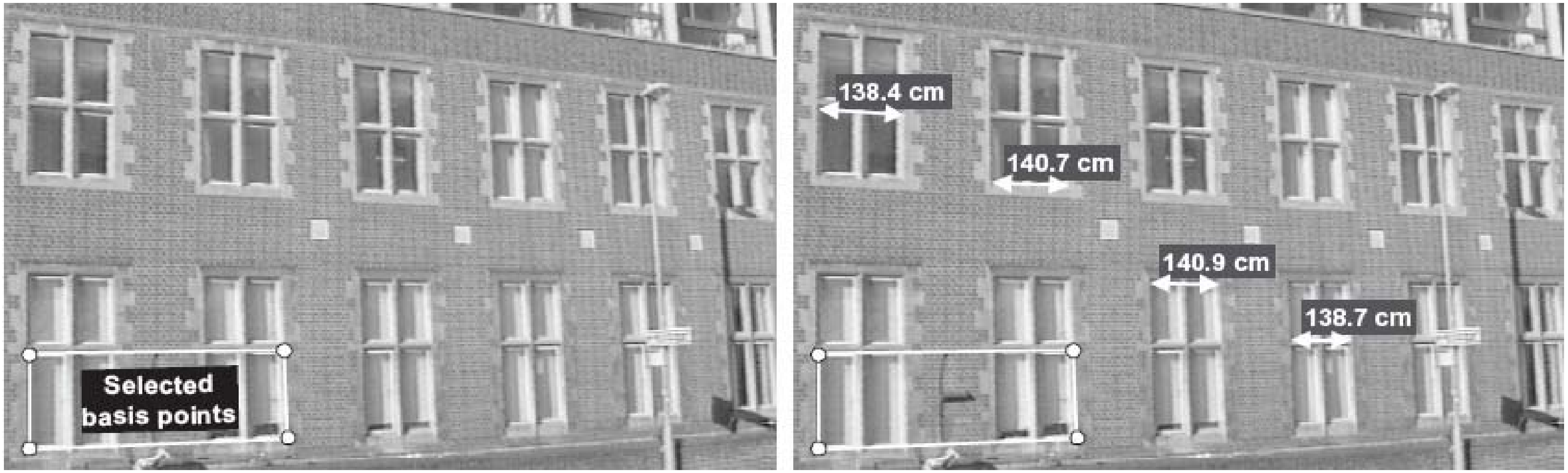
Automatic estimation of vanishing points and lines



Estimating Height

- The distance $|| t_r - b_r ||$ is known
- Used to estimate the height of the man in the scene

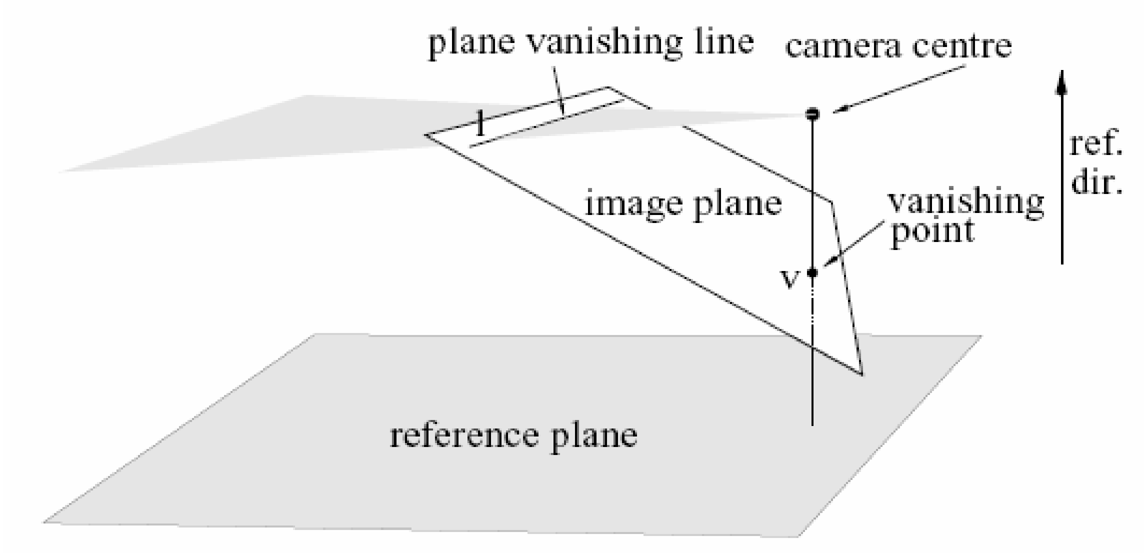
Parallel Line Segments



- Basis points are manually selected and measured in the real world
- Using ratios of lengths, the size of the windows are calculated

Camera Position

- Using the techniques we developed in the previous sections, we can:
  - Determine the distance of the camera from the scene
  - Determine the height of the camera relative to the reference plane

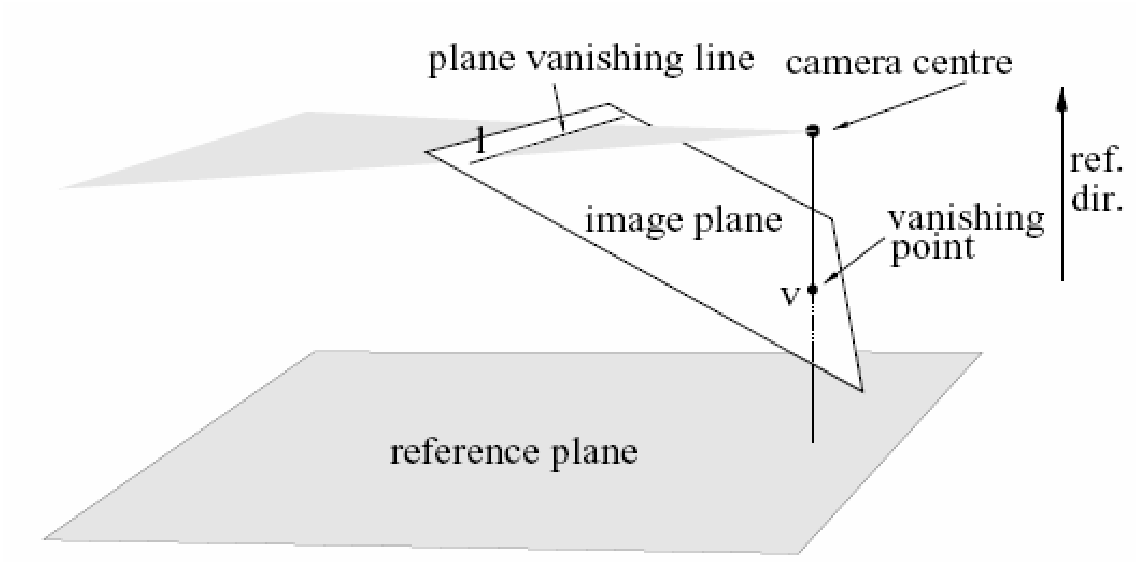


Camera Distance from Scene

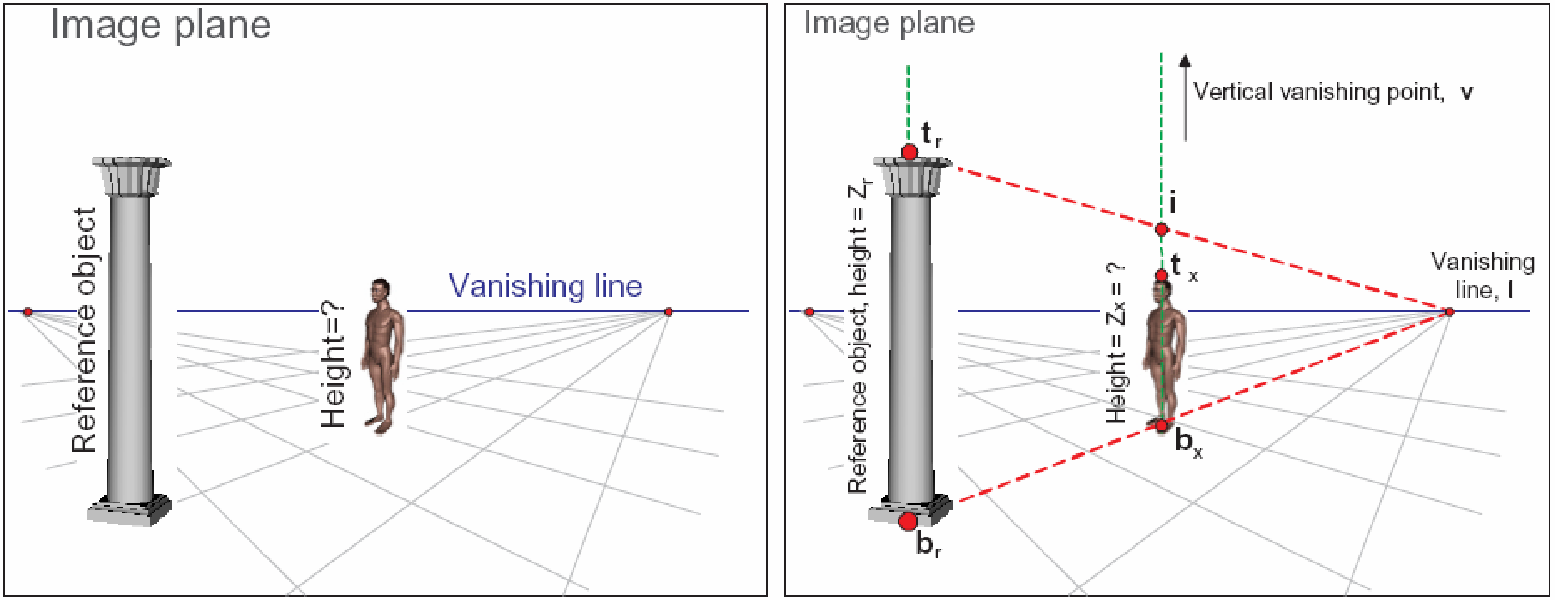
- In *Measurements between Parallel Lines*, distances between planes are computed as a ratio relative to the camera's distance from the reference plane
- Thus we can compute the camera's distance from a particular frame knowing a single reference distance

Camera Position Relative to Reference Plane

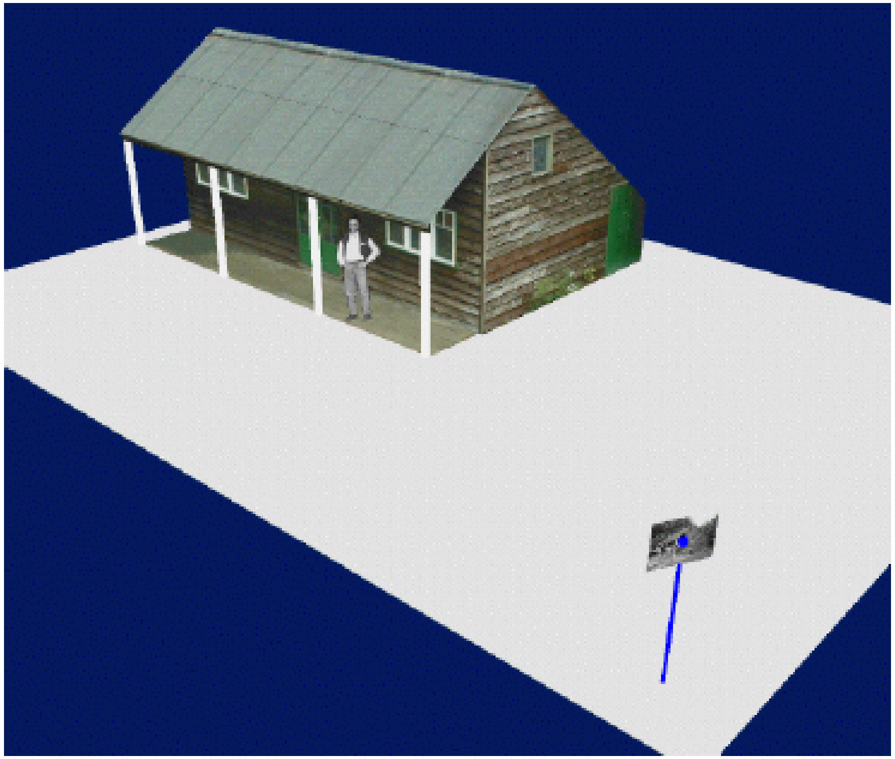
- The location of the camera relative to the reference plane is the back-projection of the vanishing point onto the reference plane

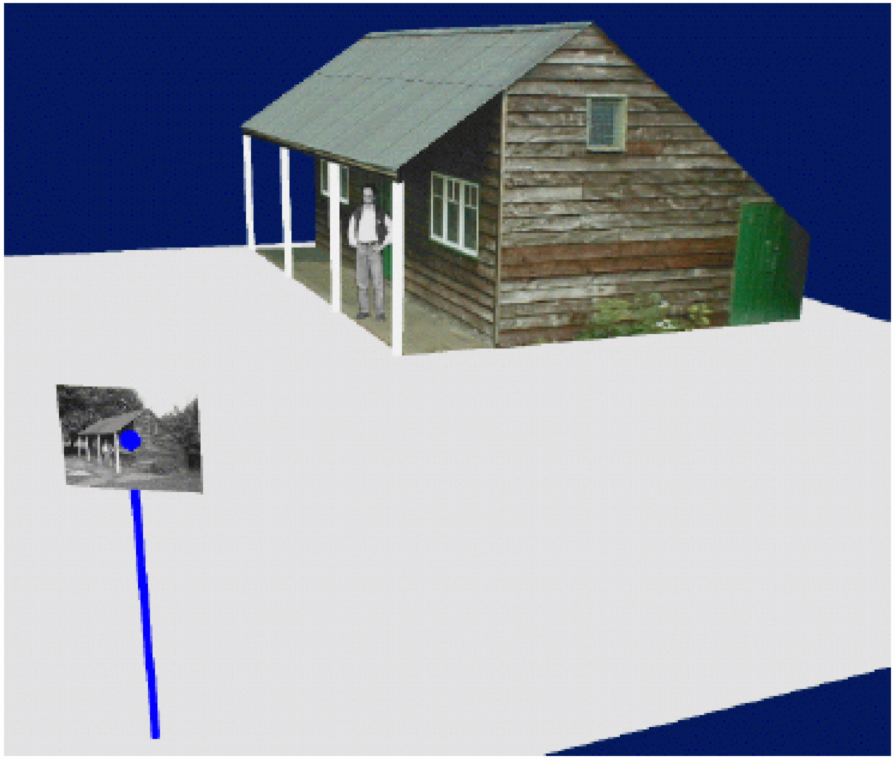


Representation



Camera In Scene





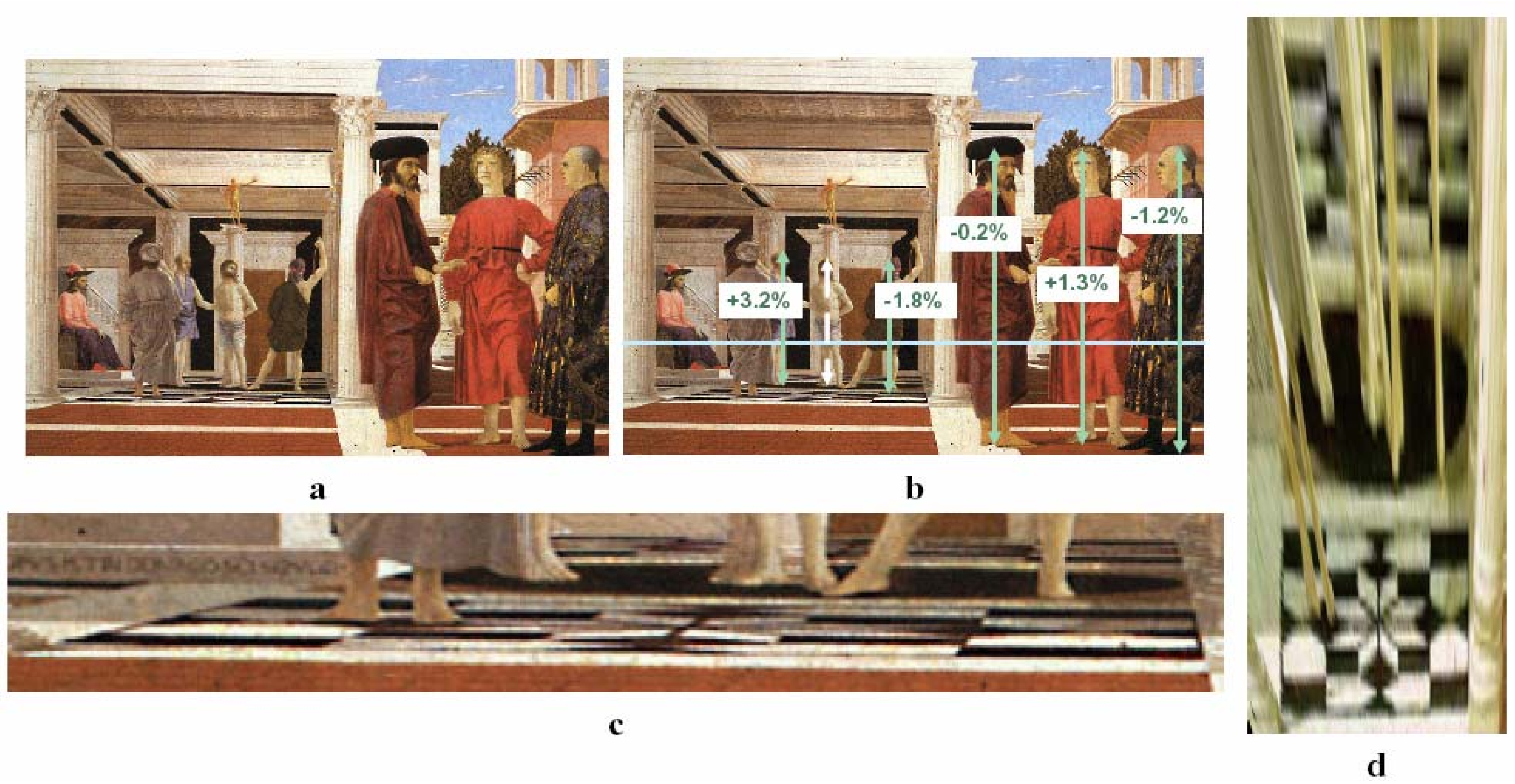
Applications

- Forensic Science (法医科学)
  - Height of suspect
- Virtual Modeling
  - 3D reconstruction of a scene
- Art History
  - Modeling paintings

Forensic Science

Virtual Modeling

Art History



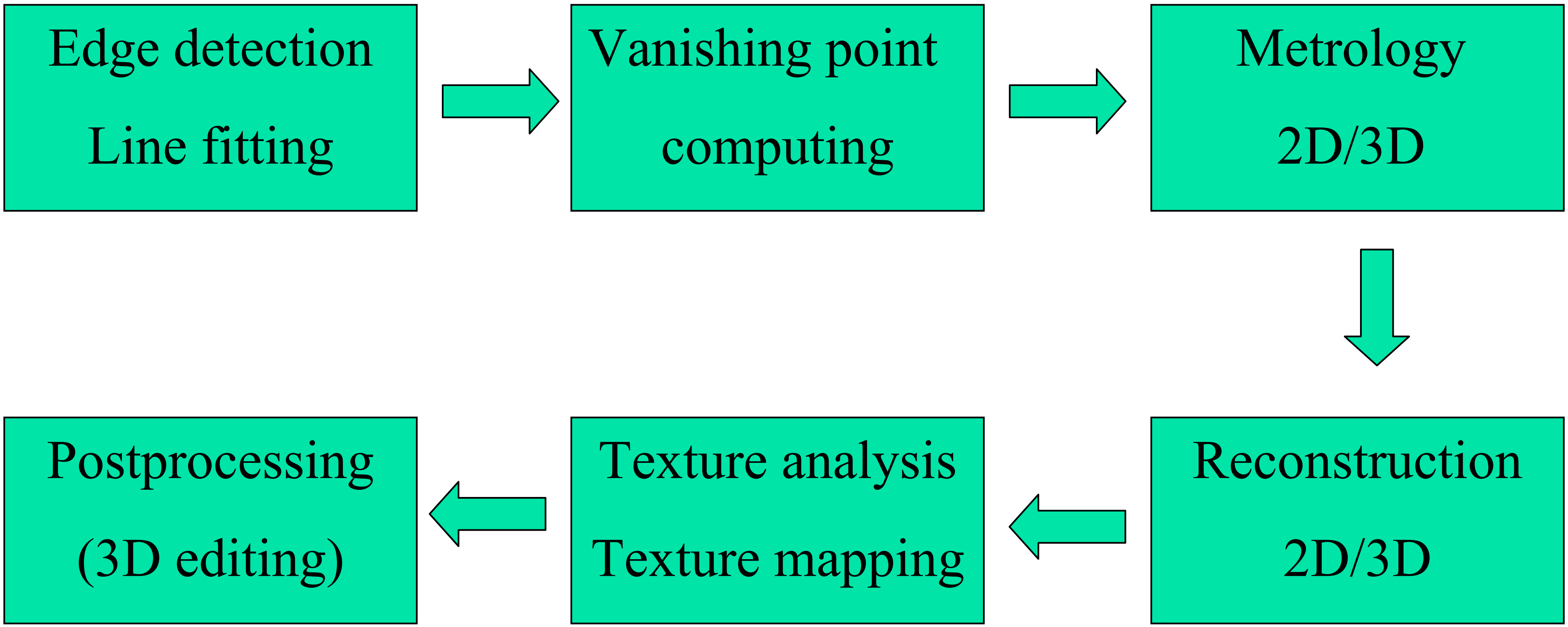
Conclusions

- Affine structure of 3D space may be partially recovered from perspective images
- Measurements between and on parallel planes can be determined
- Practical applications can be derived

Criminisi et al., ICCV 99

- Complete approach
  - Load in an image
  - Click on parallel lines defining X, Y, and Z directions
  - Compute vanishing points
  - Specify points on reference plane, ref. height
  - Compute 3D positions of several points
  - Create a 3D model from these points
  - Extract texture maps
  - Output a VRML model

A general pipeline



SingleView/Detection/fitting/calibration/metrology/optimization/
Structure/modeling/Texture/editing/…

3D Modeling from a Photograph

3D Modeling from a Photograph

Assignment/Project

- Any work related to image and vision computing is acceptable
- Presentation at the end of the semester
  - Pre-submission of demos, codes, and documents
  - PPT and DEMO at presentation
  - Each student has around 30 minutes
- Good work win exemption of final exam

Assignment/Project presentation

- June 3rd, Tuesday, 16:50-18:40
- Presentation: powerpoint slides, 30 minutes
- Presubmission (By June 1st, 23:59):
  - Codes (concise) with comments (Detailed)
    - Distinct your work from open sources (if any)
  - Result demos (Mandatory)
  - Documents (Emphasis on your own work)
- The sooner, the better.

Document/PPT format

- Abstract(摘要)
- Introduction(引言)
- Related work(相关工作)
- Your work--major part(你的工作——重点)
  - Main idea(思想)
  - Global framework / Step-by-step pipeline (整体框架)
  - Implementation details / Algorithms(实现细节)
- Experimental result(试验结果)
- Conclusion and future work(结论)
- Please refer to formal technical papers

Source code format

- Executable
- Comment whenever possible (with your name)
  - Open source: highlight your understanding
  - Your work: tell technical detail
  - Sample:
    - glCallList(iframe % Npat + 1); // chenyisong:每Npat=32帧一循环，每一帧调用一个显示列表
    - glBegin(GL_QUAD_STRIP); // chenyisong:将新的背景噪音作为源混合，为什么这里只绘制一个单位方格？
- Make your work understandable and convincible